POLITECNICO DI TORINO

Master’s Degree in Mechatronics Engineering

Master’s Degree Thesis

Self-Diagnosis Features for Estimation Models

Supervisors

Prof. Alessandro RIZZO

Ing. Giovanni GUIDA

Candidate

Hosam Elmuataz Elmansi Abdalla

April 2022

Abstract

Nowadays, developing estimation algorithms that can provide an accurate and optimal estimation of the targeted output remains to be a challenging topic for both research and industrial areas.

Despite the numerous numbers of research done on the topic of improving the efficiency and accuracy of different estimation algorithms, there is always a margin of failure that can be caused by the infinite size of the input space where it can make the algorithm unable to estimate the correct output. This problem is specifically evident during runtime since many uncertainties will rise that are caused by different working conditions and it is impossible to know if our algorithm is providing accurate estimation or not.

To address this problem, this academic work and in collaboration with Brain Technologies, proposes new state-of-the-art methodologies titled the self-diagnosis features. It provides the estimation algorithm the ability to carry out an on-line self-assessment of their performance. In addition, it can be equipped to different types of applications that rely on the estimation algorithms.

These new methodologies are based on the unsupervised machine learning algorithm of the data clustering. The main idea is to divide the input space into multiple clusters where each cluster correspond to a performance accuracy range. In this way, the expected performance of the estimation algorithm for the new data arriving during runtime can be estimated by the Euclidean distance from the cluster's centroids. The approach adopted in this thesis work is firstly to have a dissertation of self-diagnosis features, study the state of art and present the theory for the mentioned methodologies. Lastly, two studies were performed on two applications to demonstrate the practicability of the approach, which they are:

- The Estimation of the state of charge (SoC) of the battery management system
- The Estimation of the state of health (SoH) of the CNC machine

Following These studies, a conclusion shall be provided to prove the feasibility of the approach and provide a final review on the methodologies.

Table of Contents

List of Figures

List of Tables

Acronyms

SDF

Self-Diagnosis Features

FFNN

Feed Forward Neural Network

SoC

State of Charge

SoH

State of Health

CNC

Computer Numerical Control

DC

Direct Current

AI

Artificial Intelligence

Chapter 1

1 Introduction

1.1 What are the Self-Diagnosis Features?

Self-diagnosis or self-assessment is the relatively individualistic and intended engagement in reviewing and analyzing one's work in an appraisal of progress made over time. it is often considered to be closely similar to self-monitoring. It is meant to increase the learner's self-responsibility and dependency in learning and adapting [1]. The main idea presented by this thesis work was inspired by this concept of self-diagnosis and provide different estimation models for different applications the ability to behave like a human and self-assess their performance while they are deployed and on-line.

1.2 Importance of the Self-Diagnosis Features

With the constant evolution of technology and with the realization of the industry 4.0,

many efforts have been done to improve the reliability of the estimation models and algorithms. But when considering that these models are dealing with an inference space that includes an infinite number of possible input values that are affected by the various working conditions, there will always be a possibility that the algorithm might fail during runtime. In addition, the complexity of these models is increased when it is required to produce an improved accurate result. A prime example of this is the neural networks

lets say a neural network was trained on a dataset extracted from the input space and presume that the network had a high performance and confident level with this dataset. When testing this same network with a dataset from the same input space that is very different from the training dataset, there is no guarantee that the performance shall be optimal, and we will have no knowledge of this when the network is deployed to a hardware and ran on-line.

This is when the qualities of self-diagnosis come into play. It allows the neural network or any other estimating model to self-evaluate and determine if it is acting appropriately or not and understand if the readings of the estimation, we are obtaining are dependable or not.

The characteristics of the self-diagnosis features are:

- Simple and Easy to implement;
- The ability to perceive the estimation model performance;

- Functions during runtime;
- Ability to implement for different estimation models by only changing the centroids of the clusters.

1.3 The Input Space

The input space for a model is the collection of all potential inputs in vector form. The amount of possible inputs is often vast, and testing a model by feeding it all conceivable inputs is frequently impractical. In most cases, just a small number of components from the input domain are used to train and test the model. How these pieces are chosen is a significant topic in model testing. Random testing, which picks test cases at random from the full input domain, is one apparent option. [2].

However, in this thesis work, the input space shall be divided into many clusters where each cluster has its own feature differentiating it from the others.

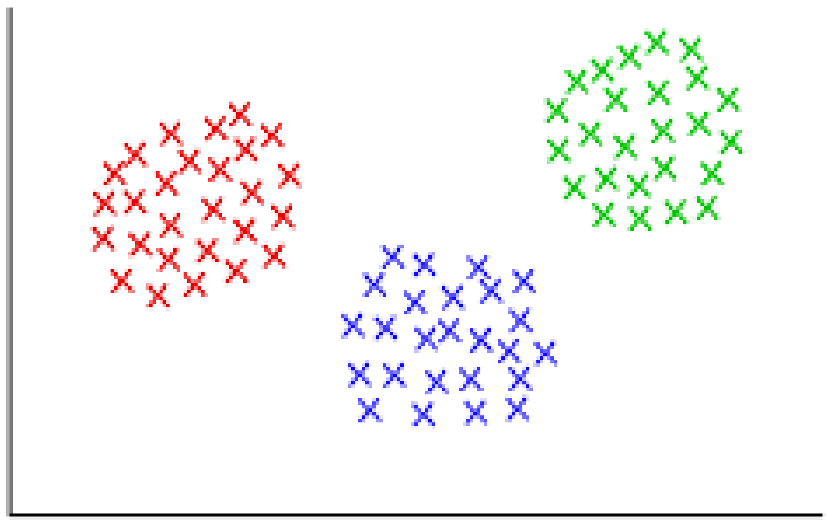


Figure 1.1 Clustered Data

This method of input space partitioning is one of the essential requirements for the self-diagnosis features and it will be discussed more in depth in the theory chapter of this work.

1.4 Thesis Objective

The examination of the machine learning world is the major emphasis of this thesis study. in particular, The study of neural networks and data clustering. The neural networks are used to construct an estimation model, which is accomplished by the training and testing of a feedforward neural network using a large dataset. Using the K-means technique, data clustering is used to create various clusters of the dataset. The centroids are retrieved once the different clusters have been created in order to construct the self-diagnosis features.

With this in mind, the main goal of this thesis work is to give two different applications with the capability to self-assess their performance during runtime, which they are:

- The estimation of the State of charge (SoC) of the battery management system
- The estimation of the State of health (SoH) of the CNC machine.

1.5 Thesis Outline

The thesis work is structured into 6 chapters:

In chapter 1, an introduction related to the concept of the self-diagnosis features is explained. Then, their importance is explored with respect to the estimation models. This is followed by a description of the input space, as well as the thesis objective.

In chapter 2, a discussion about the different subsets of AI is presented and a deep dive into the deep learning and its drawbacks is discussed with a focus on the interpretability.

In chapter 3, presents the design and the realization of the self-diagnosis features. Starting with development steps of the methodology and followed by a brief description about the feed forward neural network and data clustering. And the last part covers the workflow of the self-diagnosis features.

In chapter 4, The first study for the estimation of the SoC for BMS is performed with the results of the SDF.

In chapter 5, The first study for the estimation of the SoH for the CNC Machine is performed with the results of the SDF.

Chapter 6 is where conclusions are drawn, and future work is provided.

Chapter 2

2 State of Art

2.1 Artificial Intelligence

The definition of artificial intelligence is a hotly debated topic. It is frequently used to describe technology that display degrees of intelligence that are independent of human intellect. It is a built, artificial, or machine intelligence, and it is distinguished from natural intelligence by its very definition. AI refers to human-designed technologies that can help with complicated tasks and process information in a comparable way to humans. Image recognition (seeing), speech recognition (hearing), and natural language creation are examples of computer processes that can operate and react in human-like ways (speaking). Artificial intelligence (AI) is the artificial replication of tasks and processes that would otherwise need human intellect [3].

The notion of AI is not new, although it has lately gained prominence. The reason for this is because we only have a tiny quantity of data to make precise forecasts before. However, the amount of data collected every minute has increased dramatically, allowing us to make more precise forecasts. Along with the massive amount of data, we also have more powerful algorithms, high-end processing power, and storage that can handle such a large amount of data [4].

Within the field of artificial intelligence, there are two subsets that are considered the primary emphasis of AI:

- Machine Learning;
- Deep Learning.

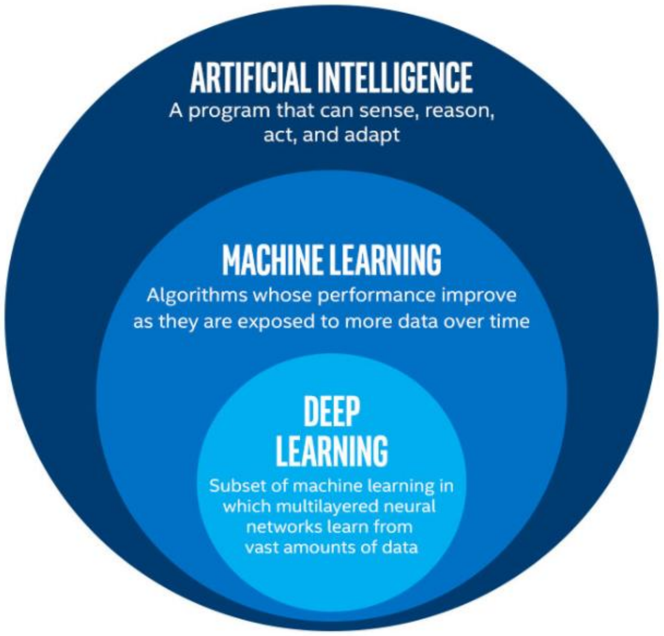

ARTIFICIAL INTELLIGENCE
A program that can sense, reason, act, and adapt
MACHINE LEARNING
Algorithms whose performance improve as they are exposed to more data over time
DEEP LEARNING
Subset of machine learning in which multilayered neural networks learn from vast amounts of data

Figure 2.1 Subsets of AI

2.2 Machine Learning

Machine Learning is a subset of artificial intelligence. It refers to a machine's capacity to acquire knowledge without being programmed. Machine learning evolved from the study of pattern recognition and computational learning theory in artificial intelligence, and it entails the research and development of algorithms that can learn from and predict data [5]. These algorithms avoid following strictly static program instructions by making data-driven predictions or decisions based on sample inputs. Machine learning is used in a variety of computing tasks where designing and programming explicit algorithms with good performance is difficult or impossible.

Supervised learning is a type of machine learning that necessitates the greatest continual human involvement, thus the name. The computer is provided training data as well as a model that is specifically meant to 'teach' it how to respond to it.

Machine learning is categorized into three groups based on the methodologies of learning:

- Supervised machine learning
- Unsupervised machine learning
- Semi-Supervised machine learning

Supervised machine learning implies the training of the machines using a "labeled" dataset, and the machine guesses the output based on the training. Many of the input data are already mapped to the output, as implied by the marked data. More specifically, we may state that we first train the machine with the input and output, and then we ask it to predict the output using the test dataset [5].

Unsupervised Machine learning differs from supervised learning in that there is no requirement for supervision, as the term implies. It indicates that the system is trained with an unlabeled dataset and predicts the output without any supervision. Models are trained with data that is neither categorized nor labeled in. The mostly commonly used unsupervised machine learning technique is the clustering which shall be discussed in depth on the following chapter.

Semi-supervised learning between supervised and unsupervised learning. It employs a combination of labelled and unlabeled datasets throughout the training phase and constitutes a middle ground between supervised with labeled training data and unsupervised with no labeled training data learning methods.

2.3 Deep Learning

Deep learning is a subset of machine learning that focuses on algorithms inspired by the structure and function of the brain [6].

Because they've been carefully patterned after the human brain, deep learning models provide an extraordinarily complex approach to machine learning and are geared to solve these issues. Data is transmitted between nodes (similar to human neurons) in highly coupled ways using complex, multi-layered neural networks. As a result, although it takes a large amount of data to be fed and create such a system, it may start producing results almost immediately, and there isn't much need for human interaction once the algorithms are in place.

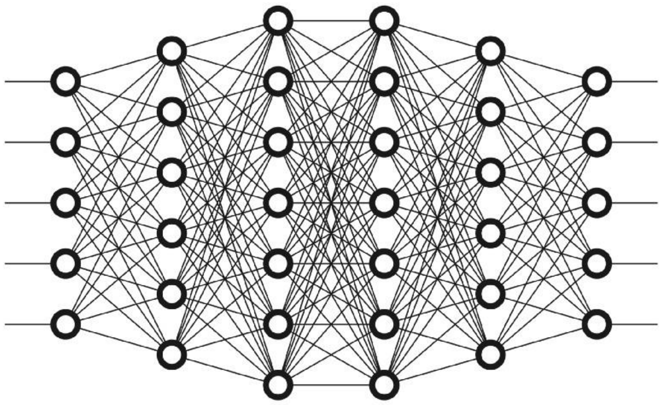


Figure 2.2 Deep Neural Network

2.3.1 Popularity of Deep Learning

Deep Learning is becoming increasingly popular as a result of its superior accuracy when taught with large amounts of data.

Deep learning has a lot of advantages, and the fact that it's fueled by vast quantities of data is a significant part of why it's becoming so popular. The "Big Data Era" of technology will open up a plethora of possibilities for deep learning innovation.

Most of the applicable features in traditional Machine Learning approaches must be identified by a domain expert in order to minimize data complexity and make patterns more obvious for learning algorithms to work. The most significant benefit of Deep Learning algorithms is that they attempt to learn high-level characteristics from data in an incremental manner. This reduces the requirement for domain expertise and the extraction of features.

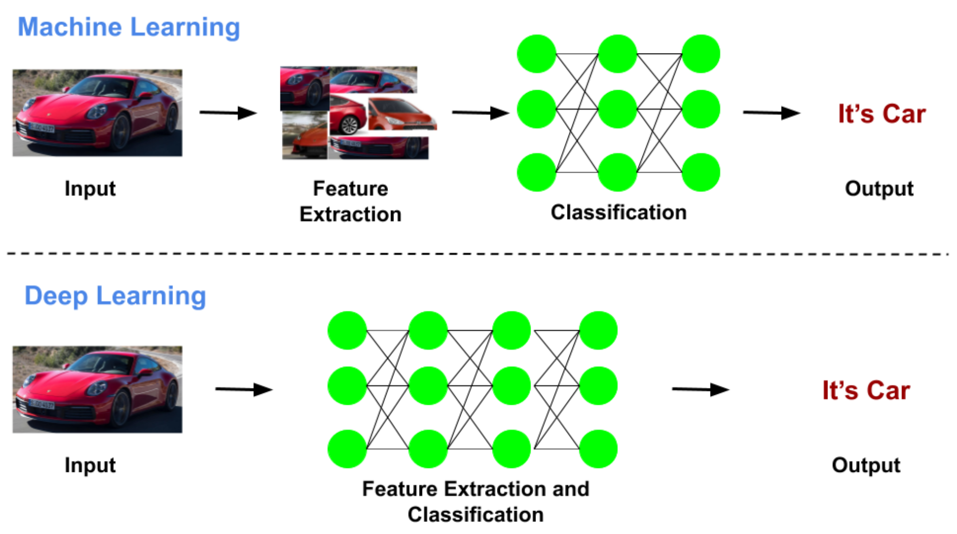


Figure 2.3 Difference Between Machine Learning and Deep Learning

The main five advantages of using deep learning are:

- Unstructured data are used to the fullest extent possible.
- Capacity to provide high-quality results
- Costs that are not essential are eliminated.
- Data labeling is not required.

Aside from its benefits, deep learning algorithms have one key disadvantage: deep learning technology cannot offer reasons for its outputs. As a result, if you don't know what the output is meant to be, evaluating the model's performance is tough. Unlike typical machine learning, you won't be able to test the algorithm and figure out why, for example, your system determined that the image is of a cat rather than a dog. This is why deep learning is considered a black box

2.3.2 Deep Learning is a Black Box

Many of the most powerful machine learning systems' features appear to be doubtful. Deep learning systems, for example, are theory agnostic in the sense that their creators do not encode a model into them that represents their grasp of the causal structure of the situation at hand. Instead, programmers create a system that "learns" a model from a massive amount of data. This design consists of layers of linked nodes that, like neurons in a brain, activate when certain characteristics in incoming data are detected. These are "deep" learners since they have several stacked levels of these nodes. In most situations, these algorithms learn when data with pre-determined classification is supplied into them. Weights on nodes in the network are automatically modified as examples collect to build the mathematical model that most properly transfers inputs to the right output labels [7].

The output classifications of such algorithms may then be compared by inputting a second batch of data whose classification is previously known. Deep learning algorithms may be trained on millions of inputs and produce incredibly accurate predictions.

despite of this accuracy, deep learning systems are categorized as black boxes. Although the architecture of these systems and the method by which they develop the models they employ for categorization are understood by their creators, the models themselves can be incomprehensible to humans. Even when procedures are employed to identify characteristics or a collection of features that a model should give considerable weight when assessing a specific case, the links between those features and the output categorization can be both indirect and brittle. A tiny change in an apparently unrelated component of the data might result in a considerable difference in feature weighting. Furthermore, differing beginning choices might lead to the creation of various models with different accuracies.

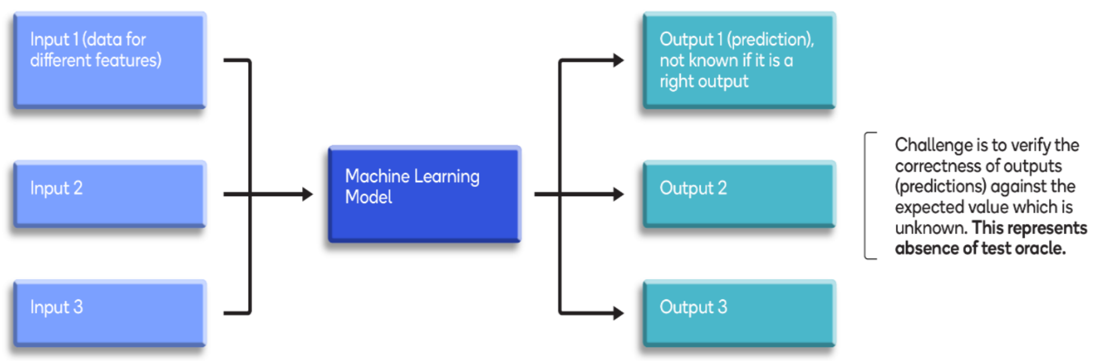

Input 1 (data for different features)
Input 2
Input 3
Machine Learning Model
Output 1 (prediction), not known if it is a right output
Output 2
Output 3
Challenge is to verify the correctness of outputs (predictions) against the expected value which is unknown. This represents absence of test oracle.

Figure 2.4 Deep Learning as Black Box

This debate about deep learning as a black box led to the idea of interpretability for deep learning and machine learning models.

2.4 Interpretability in Deep Learning Models

interpretability is the degree to which a person can reliably predict the model's outcome. The easier it is for someone to understand why particular judgments or predictions were made, the greater the interpretability of a machine learning model is. If a model's decisions are simpler for a person to understand than the other model's judgments, the model is more interpretable.

Let's take a closer look at why interpretability is so crucial. When it comes to estimation models, you must choose between knowing what is estimated or knowing why the prediction was made, and maybe paying for interpretability at the expense of estimation performance.

In certain circumstances, it isn't important why a choice was taken; all that matters is that the estimation performance on a test dataset was satisfactory. In other circumstances, though, understanding the "why" might help you learn more about the problem, the data, and why a model can fail [8].

the lack of interpretability is evident in deep learning models. Specifically, the neural networks. This can be explained by the fact that Deep learning is a black box model. Although it performs admirably in practice, its underlying mechanism and behaviors are difficult to describe.

The figure below visualizes a plot that compares the relationship between accuracy and interpretability.

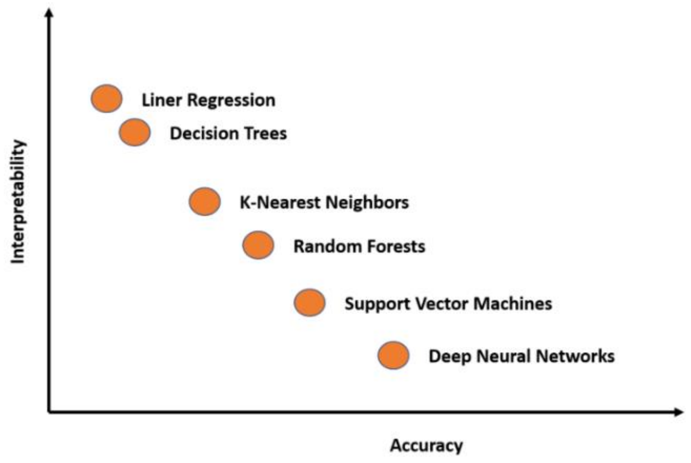


Figure 2.5 Interpretability and Accuracy

As can be seen from the graph, deep neural networks produce the most accurate answers but have the least interpretability. in other words, it can deliver very accurate results while functioning in nominal settings, but it is impossible to anticipate its behavior when working under other conditions induced by the black box phenomenon.

2.5 Interpretability and Self-Diagnosis Features

Self-diagnosis features and Interpretability of a model share the same goal of helping the estimation model to be able to understand its behavior.

The more complicated the model, the more precise it is. However, this does not imply that it will perform properly on other data. Overfitting the training data might explain for its doubtful accuracy. These models have a low interpretability and it's difficult to predict how they'll respond on data that's not the same as the training data.

The notion of self-diagnosis features came to fruition as a result of this research of limited interpretability.

Different algorithms are being produced in the industry from sophisticated models that can be significantly overfitted. This is because this model is not intended to act outside of the training data's range. However, it is difficult to manage the working conditions that might affect algorithm performance in practice.

Estimation algorithms with limited interpretability will be able to self-assess their performance when operating on nominal conditions as well as beyond their nominal range by implementing self-diagnosis features.

Chapter 3

3 Design and Realization

The goal of this chapter is to outline the overall process of creating and deploying self-diagnosis features. Feedforward neural networks and data clustering are proposed as part of the methodology. To begin, a simulation environment was created in MATLAB and Simulink to train a neural network that would be used to for estimation in the two applications of the battery management system and the CNC machine. Second, the network's performance was validated using a testing dataset. Finally, for data clustering, the same dataset used for training and testing is gathered, and the clusters centroids are extracted. Finally, the self-diagnosis features are implemented utilizing the retrieved centroids and the concept of Euclidean distance.

In detail, the development of the method followed this flow:

1. Creation of the simulated environment on MATLAB/Simulink;
2. Collecting dataset and divide it into training and testing;
3. Training a **feed forward neural network**;
4. Testing the neural network and validate its performance:
5. Extracting new variables from the dataset called as **the effective clustering features;**
6. Clustering the dataset using **K-Means Clustering**;
7. Extracting the centroids;
8. Developing the self-diagnosis features;
9. Compare results with the performance results obtained from the network.

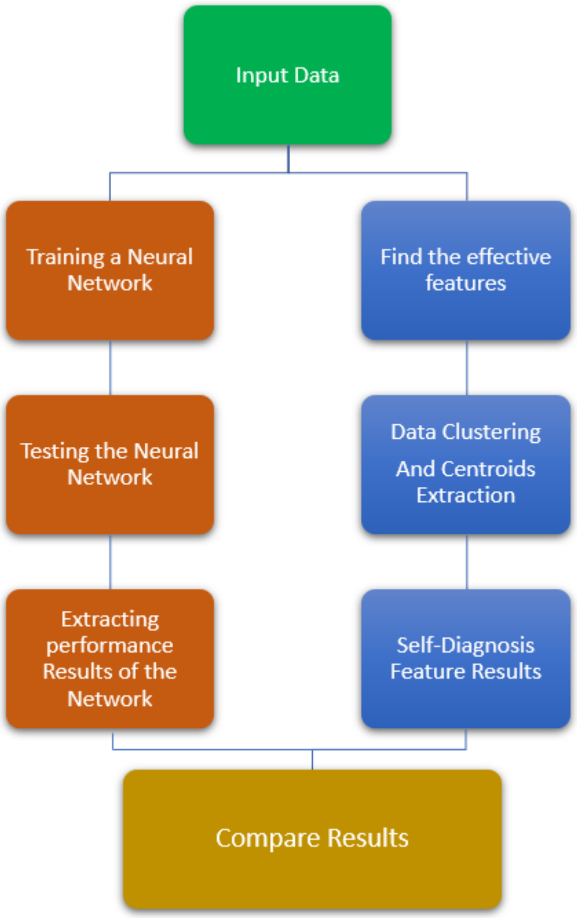

Input Data
Training a Neural Network
Find the effective features
Testing the Neural Network
Data Clustering And Centroids Extraction
Extracting performance Results of the Network
Self-Diagnosis Feature Results
Compare Results

Figure 3 Methodology for the Self-Diagnosis Features

3.1 Feed Forward Neural Networks

Looking at the previous figure, the feedforward neural network shall be used for the purpose of estimating the output values of the two case studies of the battery management system and the CNC machine. after performing the estimation, its performance is validated using the self-diagnosis features.

A feedforward neural network is a classification algorithm that is biologically inspired. It is made up of a (potentially large) number of basic neuron-like processing units that are layered together. Every unit in a layer is linked to all the units in the layer before it. These connections are not all created equal: each one may differ in terms of strength or weight. The weights on these connections represent a network's knowledge. The units of a neural network are frequently referred to as nodes.

Data enters at the inputs and flows layer by layer across the network until it reaches the outputs. There is no feedback between layers during normal operation, which is when it functions as a classifier. It's for this reason that they're known as feedforward neural networks [9].

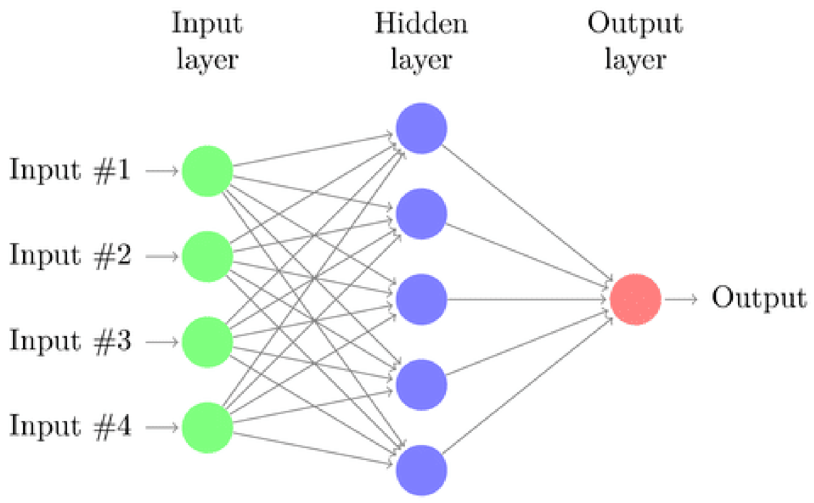


Figure 3.1 Feed Forward Neural Network

As seen in Figure 3.1, the following components make up feedforward neural networks:

- **Input layer**: This layer accept data and send it to the various layers of the neural network;
- **Output layer:** To predict the output depending on the model that is being built;
- **Hidden Layer:** The input and output layers are separated by the hidden layers. The number of hidden layers is defined by the model. many neurons in the hidden layers apply changes to the input before transmitting it. The network's weights are adjusted on a regular basis to make it more predictable;

at the core of the feed forward neural network lay three important functions:

- **Activation Function**: this is the decision-making center for the neurons output. Based on the activation function, neurons make linear or non-linear judgments.

Because it passes through multiple layers, it prevents the cascading effect from enlarging neuron outputs. The sigmoid, Tanh, and Rectified Linear Unit are the three most important activation functions [10]. In this thesis work, the **sigmoid function** shall be utilized for the output estimation of the network;

- **Optimizer:** An optimizer is a function or algorithm that alters the characteristics of a neural network. As a result, it aids in the reduction of total loss and the improvement of accuracy [11] Some are classic optimizers such as Gradient descent and Stochastic gradient descent (SGD), Gauss–Newton and Levenberg**-Marquardt** which is used in this work.
- **Loss Function:** The difference between the expected output and the outcome delivered by the machine learning model is quantified by the loss function in a neural network. We can obtain the gradients that are utilized to update the weights from the loss function. The cost is calculated as the average of all losses [12]. The most used loss function is **the Root Mean Square Error (RMSE).**

3.1.1 Sigmoid Activation Function

The sigmoid function is a mathematical function with a characteristic S-shaped curve. The logistic function, the hyperbolic tangent, and the arctangent are all examples of sigmoid functions.

Because all sigmoid functions map the entire number line into a tiny range, such as 0 and 1, or -1 and 1, one of its applications is to turn an actual value into one that may be read as a probability [13].

The most common formula used for the sigmoid function is the logistic function, and it is defined as follows:

$$d(x) = \frac{1}{1 + e^{-x}}$$

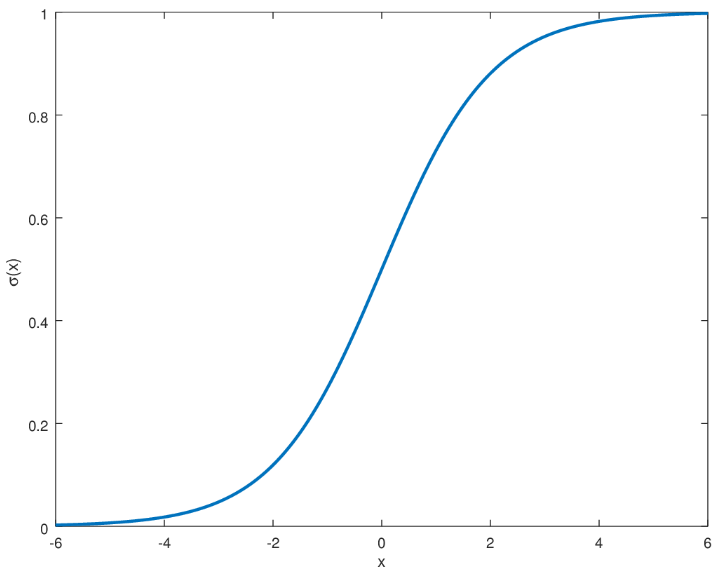


Figure 3.2 Logistic Sigmoid Function

3.1.2 Levenberg-Marquardt Optimizer

Nonlinear least squares problems are solved using the Levenberg–Marquardt (LM) Algorithm. This curve-fitting approach combines two other curve-fitting methods: gradient descent and Gauss-Newton.

Both the Gradient Descent and Gauss-Newton methods are iterative algorithms, which means they find a solution through a sequence of calculations. The gradient descent method differs in that the solution is updated at each iteration by selecting values that reduce the function value. Moving in the direction of steepest fall, in particular, reduces the sum of the squared errors. The Levenberg–Marquardt Algorithm updates the solution by choosing either gradient descent or GN at each iteration [14].

3.1.3 Cross-Validation

When using a deep learning algorithm, the optimal course of action is to train the model with different data than the testing dataset. But it is unusual to have two different dataset used for training and testing in the industrial field. This is caused by the fact that waiting for new dataset is time and cost consuming. Because of this, a random split of the dataset is performed, using a part for the training purposes and the remaining part for the sake of testing. Nonetheless, carrying out a random split of the dataset could cause biasing problems during the training process, which could lead to overfitting. This problem can be countered by performing the cross-validation. The cross-validation technique that was adopted for this thesis work was the K-fold technique,

The original dataset is equally partitioned into k subparts or folds in k-fold cross-validation. For each iteration, each group from the k-folds or groups is chosen as n testing dataset, while the remaining (k-1) groups are chosen as training data. The method is repeated k times until each group is handled as testing dataset and the remaining dataset is used for training. The technique is represented in Figure 3.3

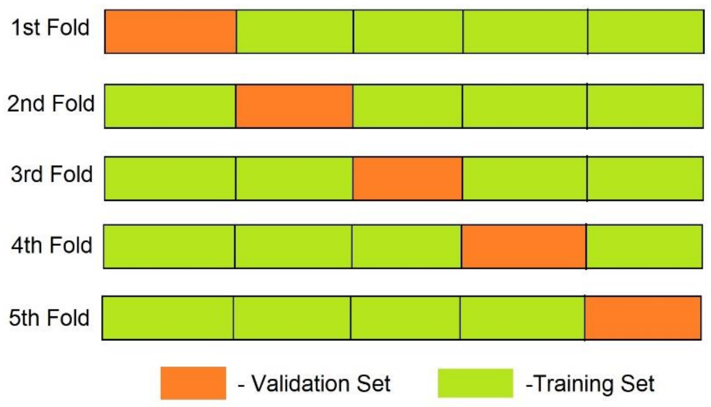


Figure 3.3 K-fold cross-validation

3.1.4 Functionality of the FFNN

The mesh of the neural network transports data. Each layer of the network functions as a filter, removing outliers and other known components before producing the final output, and the steps are as follows:

- **First Step**: Perform cross-validation for the input data;
- **Second Step**: Inputs flow through the input layer and they are multiplied by the weights;
- **Third Step**: To obtain a summation of the weighted inputs, each value is added. The output normally settles at 1 if the cumulative value exceeds the preset limit The result will be -1 if the value goes below the threshold;

- **Forth Step** : A single-layer perceptron uses the concepts of machine learning for classification;
- **Fifth Step**: With the use of the delta rule, the neural network's outputs may be compared to their expected values, allowing the network to optimize its weights through training and acquire more accurate output values. Gradient descent is the result of this training and learning process;
- **Sixth Step**: Each hidden layer is tweaked here to remain in terms with the final layer's output value.

3.2 Clustering Analysis

Cluster analysis is the process of putting objects together so that objects in the same cluster are more similar than objects in other clusters. Clusters are formed utilizing criteria such as the shortest distances, the density of data points, plus different statistical distributions. Cluster analysis has extensive applications in unsupervised machine learning.

When you have a set of unlabeled data, you're almost certainly going to use an unsupervised learning algorithm and the most effective and common algorithms are the clustering algorithms for such cases.

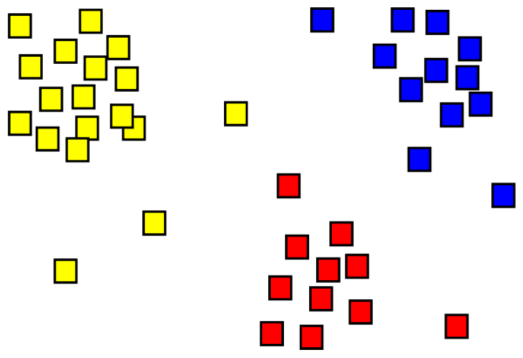


Figure 3.4 Clustered data

The algorithms used for the clustering analysis can be divided to three categories [15]:

- **Density Based Algorithms**: Data is clustered using density-based clustering when large concentrations of data points are surrounded by low concentrations of data points. Basically, the algorithm discovers and names clusters where there are a lot of data points;
- **Distribution Based Algorithms**: A distribution-based clustering approach considers all data points to be members of a cluster based on their probability of belonging to that cluster;
- **Centroid Based Algorithms**: These algorithms divide data points into groups based on several centroids in the data. Based on its squared distance from the centroid, each data point is assigned to a cluster. This is the most popular clustering technique.

In this thesis work, the focus shall be on the centroid based algorithms. Specifically, the K-Means algorithm. The reason for making such a choice is that these algorithms adopt the idea of centroids which is a major part of the development of the self-diagnosis features.

3.2.1 K-Means Algorithm

The K-Means algorithm is an iterative technique that attempts to split a dataset into K separate non-overlapping clusters, each of which contains a group of data points. It attempts to make intra-cluster data points as comparable as possible while maintaining clusters as distinct (far) as possible. It distributes data points to clusters in such a way that the sum of the squared distance between them and the cluster's centroid is as small as possible. Within clusters, the less variance there is, the more similar the data points are [16].

The way K-Means algorithm works is as follows:

- K is the number of clusters to specify.

- Initialize the centroids by shuffling the dataset and then picking K data points at random for the centroids without replacing them.
- Continue iterating until the centroids do not change.
- Calculate the total of all data points' squared distances from all centroids.
- Assign each data point to the cluster that is closest to it (centroid).
- Calculate the cluster centroids by averaging all of the data points that correspond to each cluster.

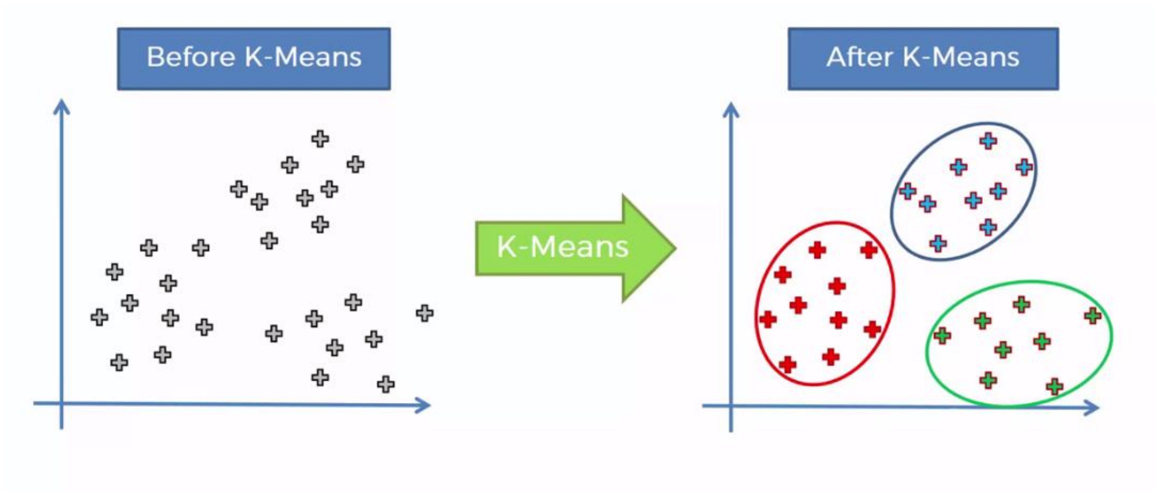


Figure 3.5 Unclustered data transformed into clustered data using K-Means

The K-Means algorithm work on minimizing the following cost function:

$$J(c_k) = \sum_{x_i \in c_k} \|x_i - u_k\|^2$$

Where:

- x_i is the point in the cluster space;
- J is the cost function;
- u_k is the mean of the data points;

- c_k is the cluster number.

3.2.2 Centroids

A centroid is a vector with one integer for each variable, each number representing the mean of that variable for the observations in that cluster. The cluster's multi-dimensional average can be thought of as the centroid. Each centroid has its own coordinates on the cluster space which it can be extracted for the purpose of developing the self-diagnosis features.

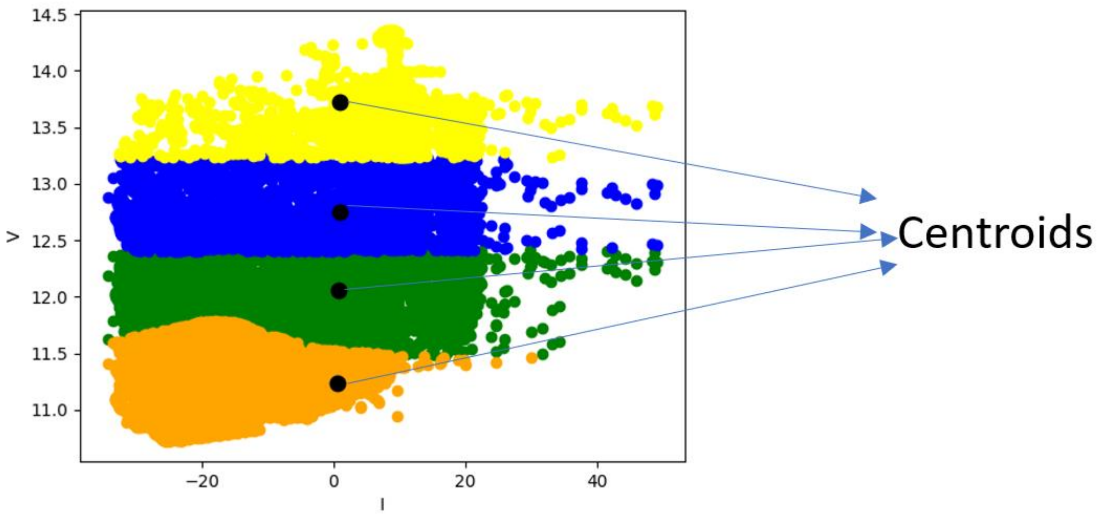


Figure 3.6 Centroids of the clusters

3.2.3 Euclidean Distance

In coordinate geometry, Euclidean distance is the distance between two points. For finding the distance between two points on a plane, the length of a segment connecting the two points is measured. We derive the Euclidean distance formula using the Pythagoras theorem.

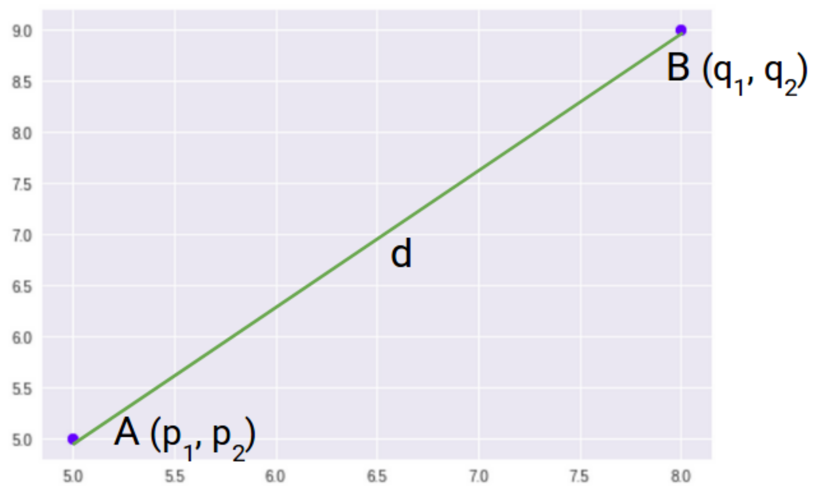


Figure 3.7 Euclidean Distance

The formula for the Euclidean distance is as follows:

$$\mathrm{d}(\mathrm{p},\mathrm{q}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

where:

- d is the Euclidean distance;
- (p,q) is two points in Euclidean n-space;
- q_i, p_i is the euclidean vectors, starting from the origin of the space;
- n is the length of the Euclidean vectors.

The Euclidean distance plays a major role in both the K-Means clustering and the self-diagnosis features. For K-Means, it is used for creating the clusters by evaluating the distance from the different centroids for the data in the space. On the other hand, for the self-diagnosis features, it is used for defining which cluster does the new data belong to.

3.3 Workflow of the Self-diagnosis Features

The objective of this thesis as discussed in 1.4, is to develop the newly proposed self-diagnosis features that provide the estimation models the ability to self-assess their performance on-line.

In this section, we shall explore the workflow of the features and the development progress. The following figure provides the development process:

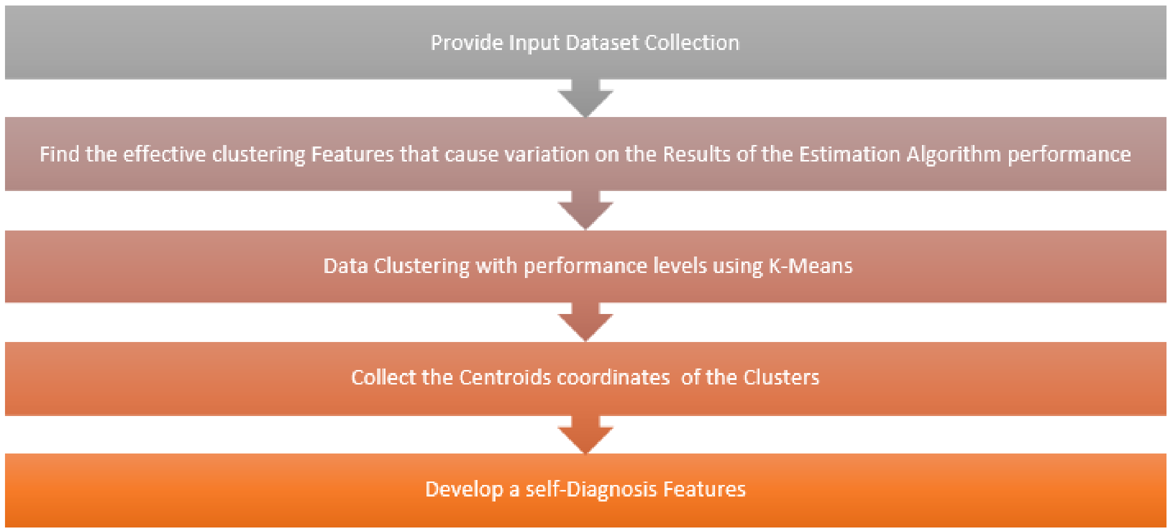

Provide Input Dataset Collection
Find the effective clustering Features that cause variation on the Results of the Estimation Algorithm performance
Data Clustering with performance levels using K-Means
Collect the Centroids coordinates of the Clusters
Develop a self-Diagnosis Features

Figure 3.8 Workflow of the Self-Diagnosis Features

Each component of the workflow for the self-diagnosis features will be briefly detailed in the sections that follow.

3.3.1 Input Dataset Collection

The first step toward obtaining the final controller is to acquire the dataset. The dataset collection process can be done on either a simulated environment or real environment.

3.3.1.1 Dataset Preprocessing

When working on a machine learning project, we don't always have access to clean, well-prepared data. It is also required to clean and prepare the data before engaging in any training or testing of our machine learning algorithm. Dataset preprocessing is the procedure for preparing raw data for use in a machine learning model. It's the first and most important stage in building a machine learning model [17].

During preprocessing, the data passes through a three major steps:

- **Data Cleaning**: Filling in empty values or eliminating rows with missing data, smoothing noisy data, or addressing data discrepancies are all examples of data cleaning;
- **Data Integration:** Data with different characteristics are grouped together;
- **Data Normalization**: Normalization is the process of ensuring that no data is duplicated, that it is all kept in one region, and that any dependencies are rational [18].

3.3.2 Effective Clustering Features

The effective clustering features are features that are extracted from the input dataset of our application. These new features can be described as a way to force a variation on the placements of all the dataset on the input space, i.e, The unclustered dataset, which in return makes it easier for the K-means algorithm to cluster the dataset into different performance levels (will be explained in details in the next section). The chosen or created features are then added to the other input features that are being used for the purpose of clustering.

The features can be obtained using **Feature Extraction** or by performing **Sensitivity Analysis.**

3.3.2.1 Feature Extraction

The process of translating raw data into numerical features that can be processed while keeping the information in the original data set is known as feature extraction. It produces better outcomes than applying machine learning to raw data directly. [19]

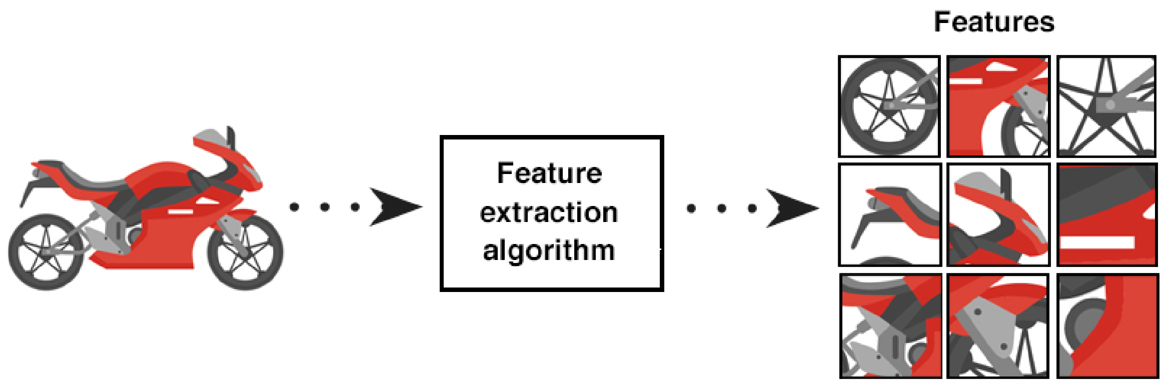


Figure 3.9 Feature Extraction

Feature extraction can be done manually or automatically, as follows:

- Manual feature extraction follows the process of Identifying and characterizing the features that are significant for a specific situation, as well as designing a method to extract those features. In many cases, knowing and having a good knowledge of the domain can assist in making educated selections about which features can be valuable.
- Automated feature extraction uses specific algorithms or deep learning networks for extracting the features automatically from the dataset without requiring human intervention [19].

For our case study of the BMS, the manual feature extraction will be applied to obtain a new effective clustering feature that shall be helpful for the development of the self-diagnosis features.

3.3.2.2 Sensitivity Analysis

The Sensitivity Analysis (SA) is a method in a numerical model that tests how the influence of uncertainties in the input variables might lead to uncertainties in the output variables. This study is beneficial in the sense that it let us visualize which variable can cause the machine learning algorithm to drop in performance. Hence, we can different performance levels in the clusters.

There are to types of sensitivity analysis (SA), global and local. The behavior of input parameters on the change of the model output is the focus of global SA. Different factors, in fact, have varying (and sometimes drastic) effects on the system's output. Because certain criteria have a major impact while others have a minor impact, global SA is a useful tool. The relative sensitivity of a single parameter value to changes in other parameters is measured by local SA.

In the case of our studies, the global sensitivity analysis is used for the application of the SoH estimation of the CNC machine, where different input variables shall be analyzed to force a variation on the accuracy of our machine learning model.

3.3.3 Data Clustering using K-Means

As presented before in section 3.2, K-Means clustering divides the dataset into multiple clusters. In the case of our development process of the self-diagnosis features, each of these clusters include samples of the dataset that when tested on the machine learning model of our application, the model provides similar accuracy for the output, or the accuracy can be in a specific range, for example, the accuracy can be in the range of 80% to 100% on one of the clusters. These clusters are called **Performance Levels,** where each performance level includes a range of the model accuracy. The figure below visualizes the performance levels

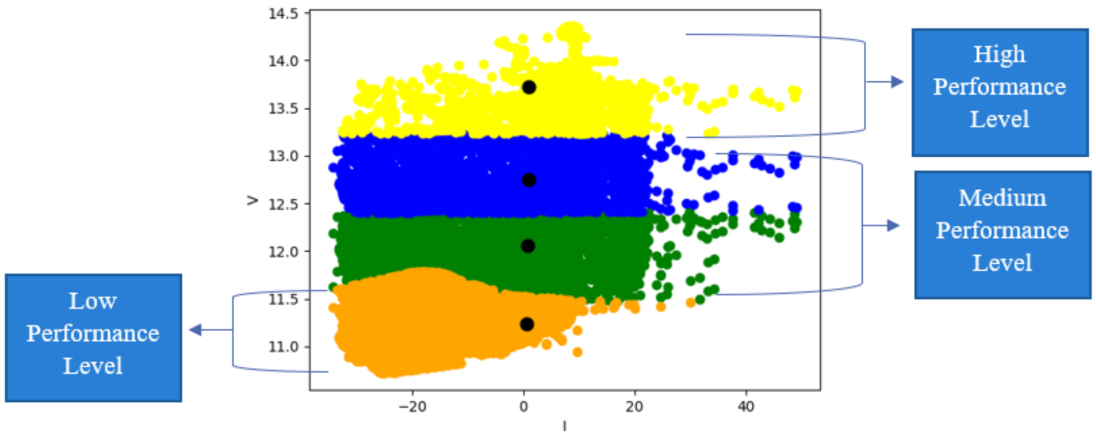


Figure 3.10 Performance Levels

To make it more understandable, lets say we have a new sample , a 2-dimentional vector x that belongs to the above dataset space in the figure:

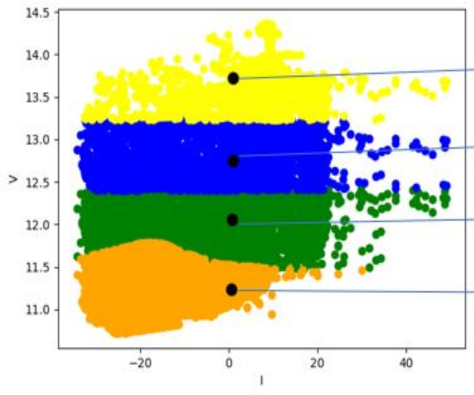
- If x belongs or it is near to the top cluster (yellow colored), then the machine learning model provides a high accuracy (81% - 100%) for this new sample;

- If x belongs or it is near to the two cluster in the middle (blue and green colored), then the machine learning model provides a medium accuracy (61% - 80%) for this new sample;
- If x belongs or it is near to the bottom cluster (orange colored), then the machine learning model provides a low accuracy (0% - 60%) for this new sample;

Also, it can be noted that two clusters can be included in one performance level as seen in Figure 3.10. The reason for this is To simplify the self-diagnosis features developing phase and focus only on a maximum of three performance levels (Low, Medium, High).

3.3.4 Centroids Extraction

As discussed in section 3.2.2, each cluster has a centroid assigned after the utilization of the K-Means algorithm. From the cluster space, these centroids are recovered as a coordinates matrix. The coordinates of the centroid for each cluster are listed in each row of this matrix, and columns represents the corresponding



```
[[12.03864703 12.03740688]
 [13.72334277 13.71678434]
 [12.74664371 12.74533693]
 [11.21253247 11.2196399 ]]
```

Figure 3.11 Extracted Centroids

3.3.5 Self-Diagnosis Features Development

The last step of the workflow is the development of the self-diagnosis features. The features depend completely on the extracted centroids of the clusters and the concept of the Euclidean distance that was explained previously. The developed features work similarly to a regular classification algorithm where it assigns any new sample of data with a performance level. Each time a new sample data is inputted to the features algorithm, the Euclidean distance from all centroids is calculated and smallest distance defines which performance level does this new data belongs to. If the new data falls outside any of the cluster's range, it will be assigned to the closest cluster.

The functionality of the features algorithm can be described in the following steps:

1. Provide new data;
2. Calculate the Euclidean distance from the centroids of all the clusters;
3. Choose the smallest distance from all the clusters;
4. Provide the corresponding performance level.

Chapter 4

4 Self-Diagnosis Features for SoC Estimation of BMS

This chapter will present the case study of the *SoC estimation of a battery management system* to which the **Self-Diagnosis Features** are applied. Firstly, a summary about the battery management system is presented. Secondly, the estimation process of SoC is performed using the feed forward neural network for the BMS. Thirdly, the implementation of the SDF is shown. And lastly, the estimation performance is tested and compared with the self-diagnosis features results.

4.1 Battery Management Systems

On the current industry, one of the most interesting topics is the possibility of controlling the production of natural resources as efficiently as possible, e.g., electricity. In the field of electrical and hybrid vehicles, the most proper aspect is the aspect of energy storage systems.

An ever need for more efficient performance, as well as the increasingly strict rules imposed as a result of safety concerns, with the constant incrementation of the costs, has led to the idea of developing management systems that provide optimization and monitoring the electricity storage systems. Those systems are what called the battery management systems BMS.

The main responsibilities of the BMS are:

- The estimation of the state of charge SoC of the battery;
- The estimation of the state of Health SoH of the battery;
- Managing the different cells of the battery.

The estimation of the State of Health SoH and the State of Charge SoC are crucial aspects for the optimal management of the entire BMS, as can be observed from the early debates on the subject [20].

The state of charge (SoC) is a percentage assessment of the amount of energy available in a battery at a given point in time. The SOC reading for a computer, for example, could be 95 percent full or 10% full. The SOC tells the user how long the battery can last before it needs to be charged or replaced. Understanding the condition of charge is critical since knowing the remaining capacity can aid in developing a control strategy [21].

The SoC depends on different factors:

- The supplied current;
- The temperature;
- The State of Health.

On the other hand, the state of health (SoH) is a parameter that provides the real conditions of the battery with respect to the ideal ones.

The SoH depends on :

- Age of the battery;
- Usage in abnormal conditions;
- Manufacturing flows.

In our study here, the focus shall be **the estimation of the SoC** and applying the SDF to its estimation algorithm.

4.2 Battery Model

The battery model that shall be used for the purpose of this case study is an equivalent circuit as shown on the figure:

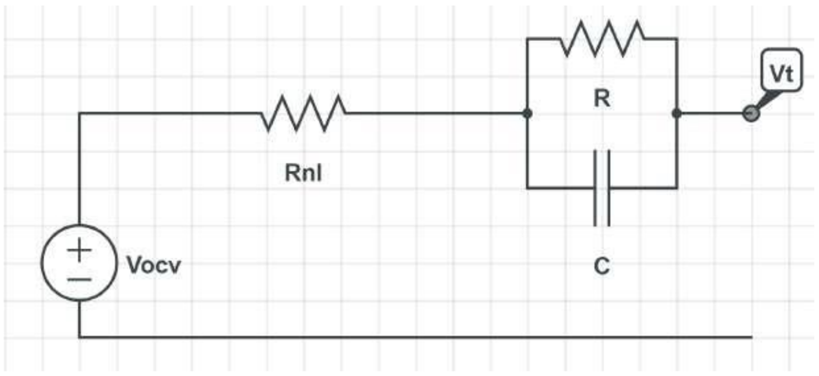


Figure 4.1 Battery Model [3]

It is made out of three main components, an open circuit voltage Vocv, a nonlinear resistance Rnl and an RC branch that represents the battery dynamics.

Since our studies are done in a simulated environment, below is the Simulink structure for the battery model that was provided by Brain Technologies and was used on the BAT-MAN project.

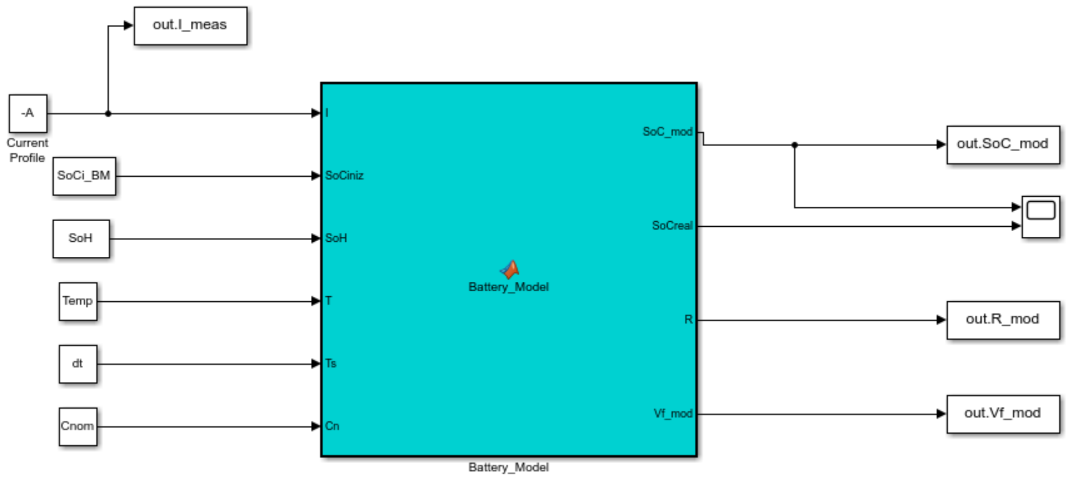


Figure 4.2 Simulink Structure for the Battery

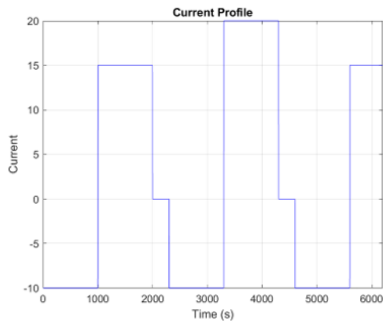
4.3 Estimation of the SoC using a FFNN

For the sake of estimation of SoC, the data generation was performed using the Simulink structure shown on the Figure 4.2, because of the non-availability of real data since it takes a long time and work to produce, which contradicts with time to produce this thesis work. It is evident that the approach shall not be as a real implementation, but it can be a head start for the future development of the SDF.

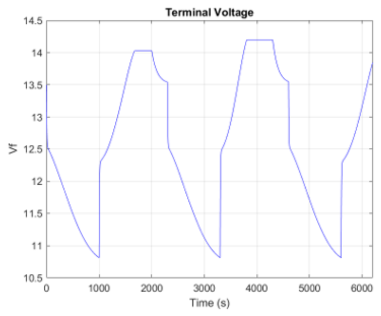
Keeping this in mind, the data collection process was done using the Simulink structure at Figure 4.2. Three simulated sensors were placed with the purpose of retrieving three variables from the model, the current, the terminal voltage and the SoC.

The battery that was used for the sake of this study was considered to always be at SoH 100%, with a nominal capacity of 4.2Ah and a constant temperature of 25^{o}.

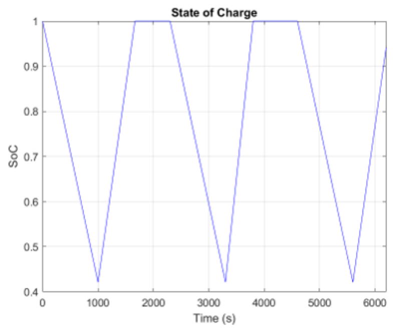
For the estimation that will be performed, the current profile which was used is shown in the figure below with the corresponding voltage and the SoC behavior.



Current Profile



Terminal Voltage



State of Charge Corresponding to the Current Profile

Figure 4.3: The Current Profile used with the corresponding voltage and SoC behavior

The current profile has the following features:

- Constant current and constant voltage cycles;
- Step charge and discharge cycles;
- two cycles for allowing the battery to charge/discharge multiple times;
- Peak current of +15/-10 to allow the fast charge/discharge of the battery.

4.3.1 Training and Test Dataset

Following the choice of the current profile, we procced to create the different dataset that will be used to train and test the feed forward neural network, where its structure was presented in 3.1.

The dataset is created by using the previous current profile by only varying the initial state of charge SoCi (which is the initial value of the SoC during runtime). For the training dataset, the SoCi is considered at 100%, which is the nominal value for the battery. On the other hand, for the test dataset, the SoCi is varied with equally spaced interval such that SoCi = [20%, 40%, 60%, 80%].

4.3.2 FFNN Specification

After performing data pre-processing and randomizing the dataset using the K-fold cross-validation as explained in 3.3.1.1, the next step is the development of the FFNN that shall be used for the purpose of the estimation of the SoC. The Inputs to the network are the current and voltage, while the SoC is the output.

The chosen FFNN had the following specifications as shown on the following table:

Specifications	FFNN
Number of inputs	2 (Current, Voltage)
Number of neurons in the hidden layer	14
Number of outputs	1 (SoC)
Activation Function in hidden layers	Sigmoid
Optimizer	Levenberg-Marquardt
Loss function	RMSE

Table 1 FFNN Specification

The choice for the number of neurons hidden layers is done by using the loss function of the *root mean square error* RMSE, as mentioned in 3.1 . That is done by training the network with different number of neurons ranging from 1 to 30. After training the network 30 times, we choose the optimal number of neurons for the hidden layers by finding the lowest value of RMSE for both the training dataset and the validation dataset (which is 20% of the training dataset).

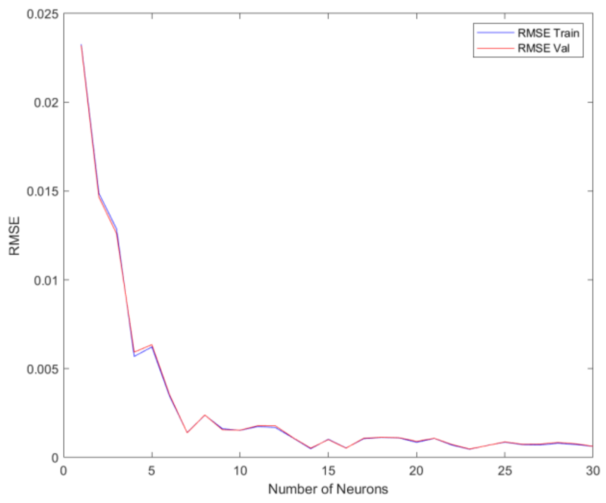


Figure 4.4 Relation between RMSE and the Neurons

As it is visible from the plot in Figure 4.4, the optimal choice of the number of neurons is 14, where it has a value of RMSE = 0.0005.

4.3.3 Training and Testing the FFNN

Following the choice of the specifications for the network, the training process is performed with the SoCi at 100%. The plot below illustrates the output of the network compared to the actual output.

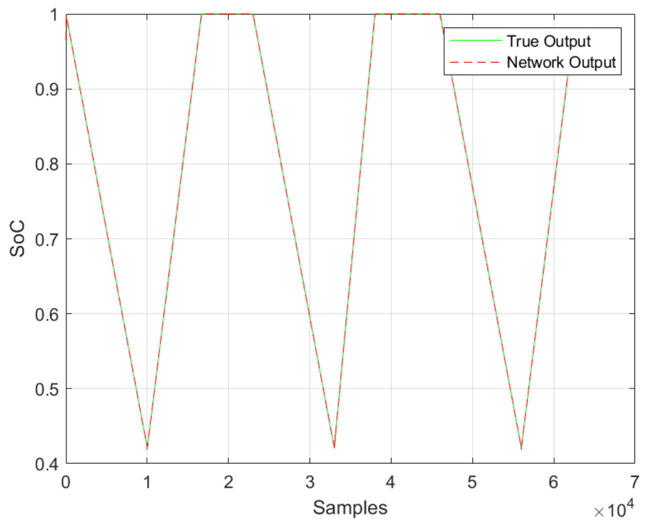


Figure 4.5 Network estimation of SoC

The network can estimate the SoC perfectly compared to the actual SoC obtained from the simulated environment, meaning that the network was able to perform optimally with training dataset (SoCi 100%).

The next step is to test the FFNN. The boxplot (see further information about the boxplots in Appendix A) in Figure 4.6 represents the accuracy of the network output at different values of SoCi.

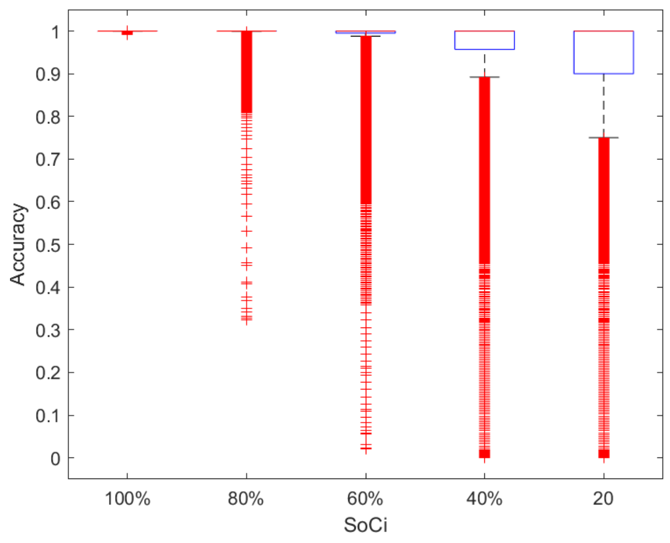


Figure 4.6 Performance of the FFNN output at different SoCi

As it is noticeable from Figure 4.6, the network is able to perform adequately in most cases for different SoCi levels. But it is also evident that the number of outliers increases as the SoCi reduces. Which means at low values of SoC, the network does not provide the desired output, which might not be acceptable in the case this network is deployed to the field. However, this is optimal case for our thesis work, since we are trying to allow the network to self-assess its performance in all cases, whether it is when the network is providing accurate results or the contrary.

4.4 SDF Implementation for BMS

The subsequent phase after extracting the performance data of the network, is to implement the SDF. When the SDF is applied to the network, we will be able to evaluate its performance and compare it to the results acquired in the previous stage.

4.4.1 Data Clustering for SoC

The first step in the implementation process is to perform data clustering using the same dataset that was used for training and testing the neural network.

The Data is clustered with respect to the current and the terminal voltage since they are the only two available inputs in the dataset.

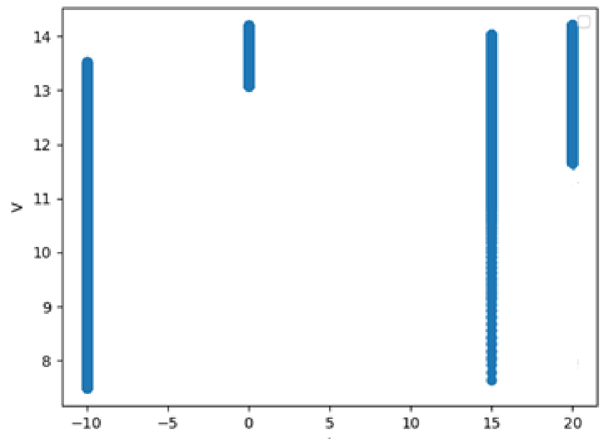


Figure 4.7 Unclusterd Dataset

Figure 4.7 illustrates the unclustered dataset. All of the dataset represents lines at different values of the current (x-axis), which is expected since the behavior of the current profile for the battery ranges between a minimum number of values.

Now, we procced to perform the K-Means data clustering and visualize the different clusters as can be seen in Figure 4.8 and it is compared to the actual accuracy of the network performance that was extracted in Figure 4.6, but this time it is visualized in the cluster space for the sake of comparison in Figure 4.9.

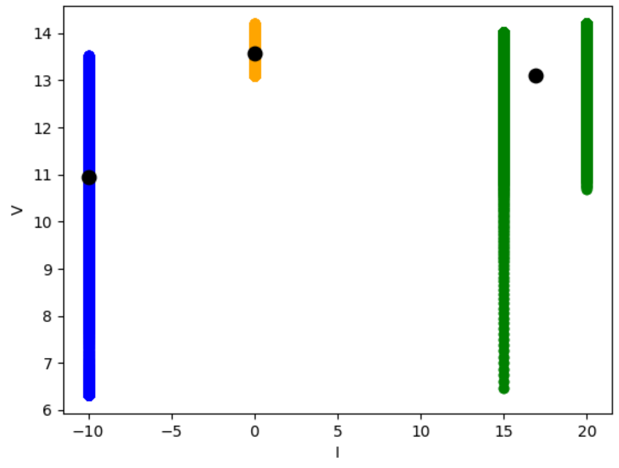


Figure 4.8 Clustered Dataset with K-Means Algorithm

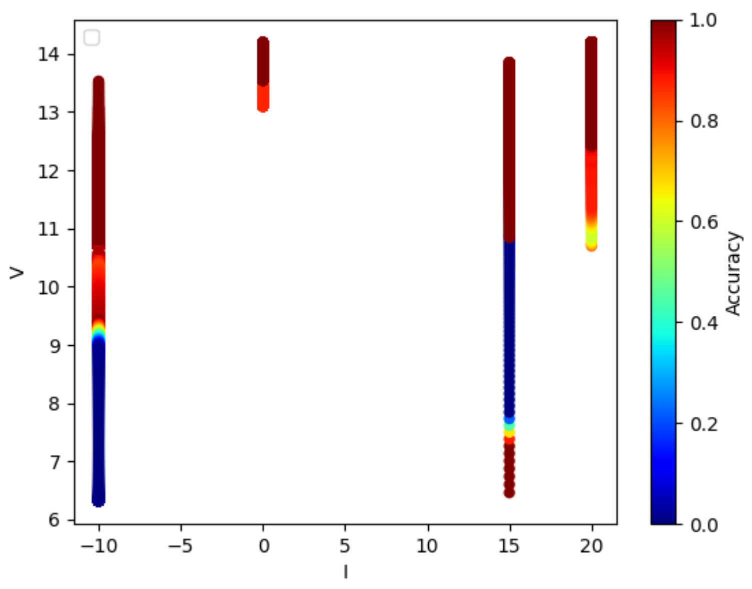


Figure 4.9 Clustered Dataset with the known Accuracy

The main objective of the K-Means algorithm for the development of the SDF is to divide the dataset into three performance levels that represents the actual accuracy of the neural network. But when looking at Figure 4.8, the K-Means algorithm is unable to capture those levels, where it is only clustering the dataset by finding which data is the closest to each of the three centroids. Comparing the results of the K-Means to Figure 4.9, it is evident that with each value of the current (x-axis), the accuracy changes, while in K-Means, two current values share the same clusters. To counter this problem, we procced to obtain an **Effective Clustering Feature.**

4.4.2 Feature Extraction for BMS

The effective clustering feature (Explained in Part 3) shall be obtained by performing the **Feature Extraction**.

The new created feature is extracted from the terminal voltage for all the available datasets. The feature is called **The Voltage difference,** which can be simply described as the difference between the maximum value of the terminal voltage in all available dataset and the newly inputted value of the terminal voltage or any terminal voltage value available in the dataset.

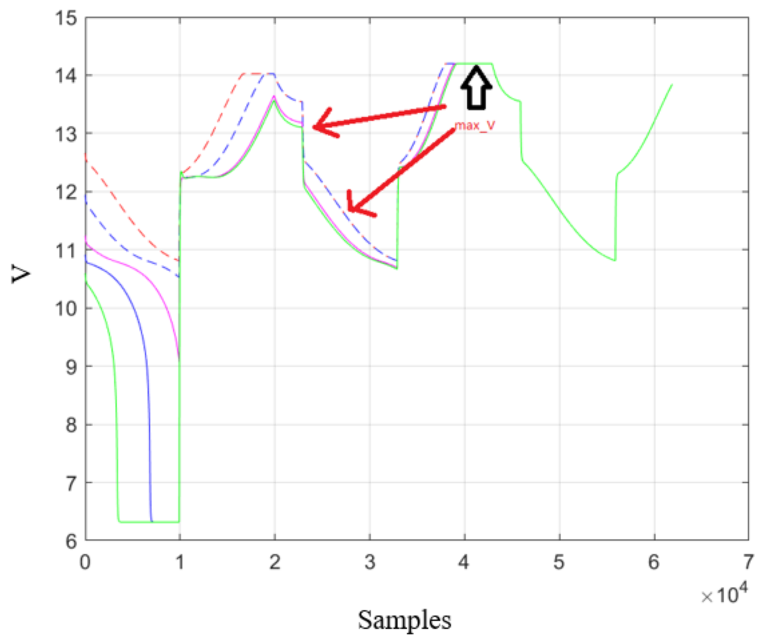


Figure 4.10 Maximum Value of the voltage available on the dataset

The above figure represents the maximum value of the voltage available on the dataset, which is $V_{max} = 14.1$.

4.4.3 Data Clustering using the Effective Clustering Features

Following this, the voltage difference is added as a new variable to the dataset and then the data clustering is performed again. This time, the K-Means clustering is done only with respect to the terminal voltage and the newly introduced voltage difference.

The figure below shows the new unclustered dataset.

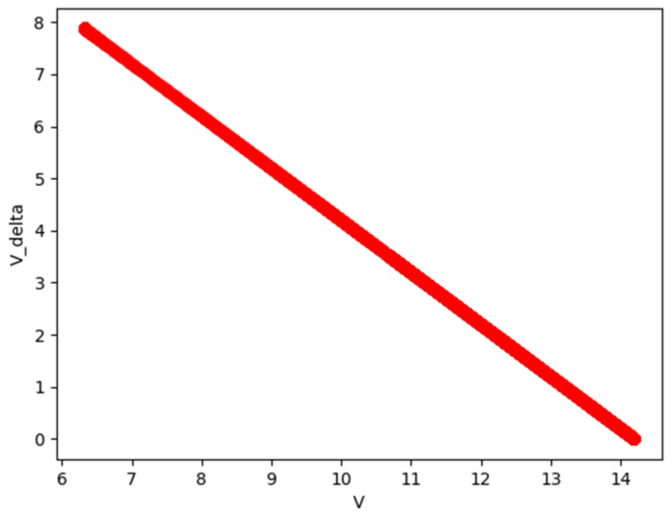


Figure 4.11 Unclustered Dataset

From the figure above, the unclustered dataset is basically just a linear function , which shall simplify the clustering process for the K-Means algorithm as seen in Figure 4.11 .

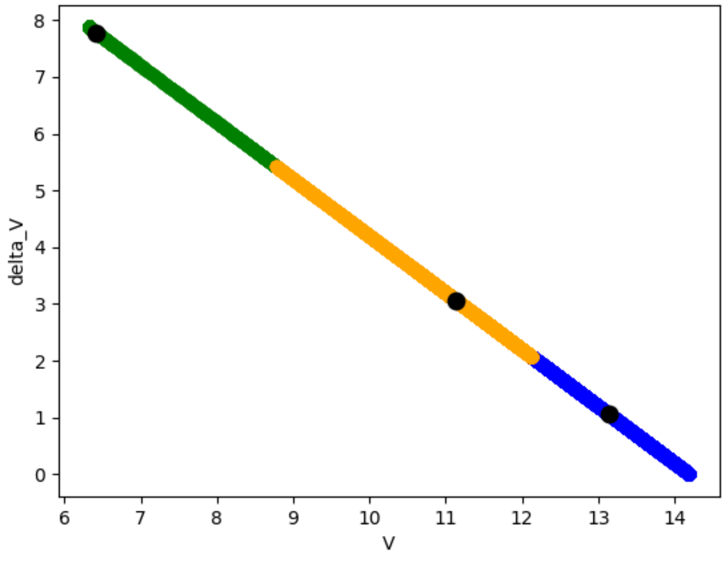


Figure 4.11 Newly Clustered Dataset with K-Means

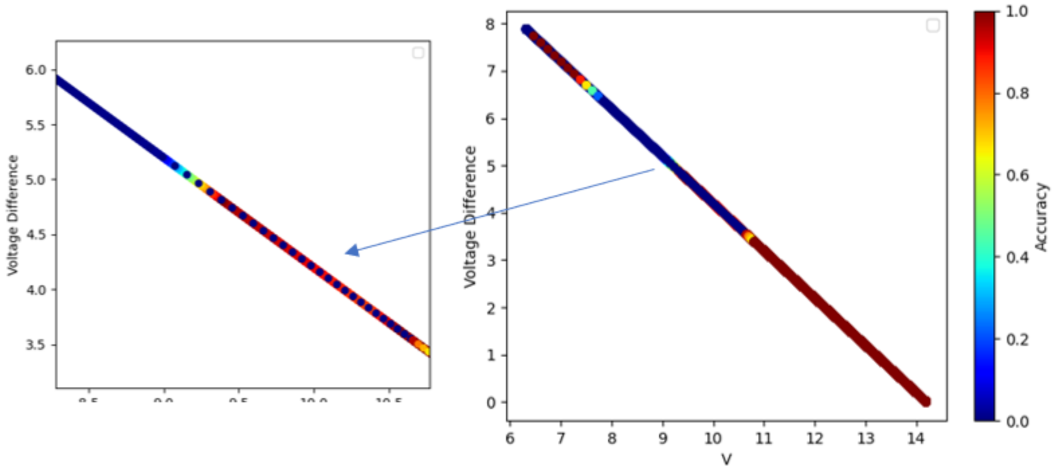


Figure 4.12 New Clustered Dataset with the known Accuracy

Looking closely at Figure 4.11, the K-Means clustered the dataset into three different clusters with each includes their centroids. When comparing the results of K-Means with the actual accuracy results Figure 4.12, we notice there is a similarity on the division of the dataset, the data on the lower part with red color of the line in Figure 4.12 represents the data with high accuracy which correspond to the blue cluster, the data in the middle (not visible on the right graph but can be seen clearly on the zoomed graph on the left in Figure 4.12) represents the data with medium accuracy which correspond to the orange cluster and lastly the green cluster correspond to the data with low accuracy with blue color in Figure 4.12.

After performing an evaluation test for the clustering of K-Means algorithm, it was found that it is capable of finding the correct match-up with an accuracy of 87.2%.

4.4.4 Performance Levels and Centroids Extraction

Knowing all this, the performance levels (explained in 3.3.3) can now be determined, which they are:

- **High Performance Level**: Includes all dataset that has an accuracy in the range of 100% to 81%, represented by the blue cluster;

- **Medium Performance Level**: Includes all dataset that has an accuracy in the range of 80% to 61%, represented by the orange cluster;
- **Low Performance Level**: Includes all dataset that has an accuracy in the range of 60% to 0%, represented by the green cluster.

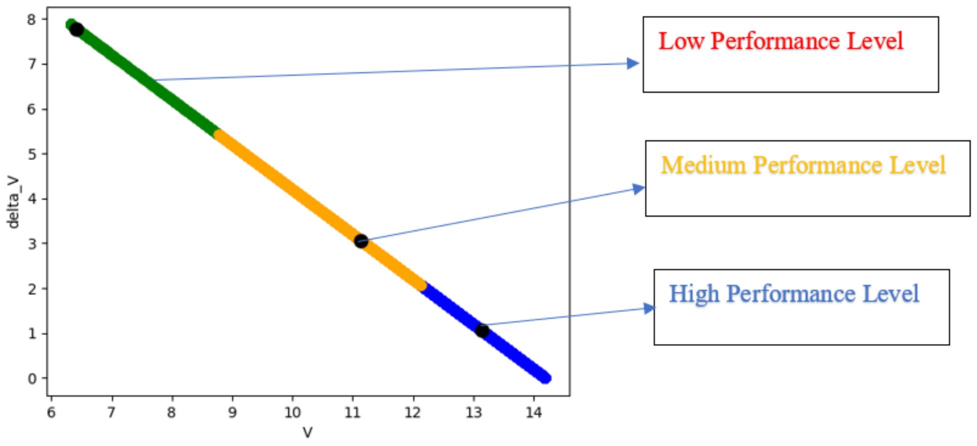


Figure 4.13 Performance Levels

Now, the last procedure before going to test the SDF is to extract the centroids coordinates from the cluster space.

The figure below shows the centroids vector of coordinates for each of the three clusters.

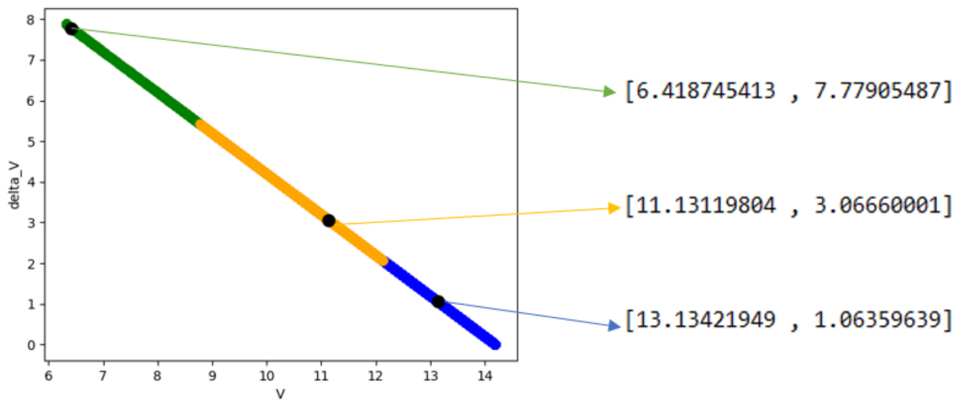


Figure 4.14 Centroids Coordinates for BMS Clusters

4.5 Testing the SDF

The last part of this case study is to perform the testing using the extracted centroids and the concept of the Euclidean distance.

In the next page, the Simulink implementation with the SDF is shown. The turquoise block is the battery model, blue block represents the SDF while the yellow block is a constant block that includes the maximum voltage obtained during the feature extraction (see Feature Extraction)

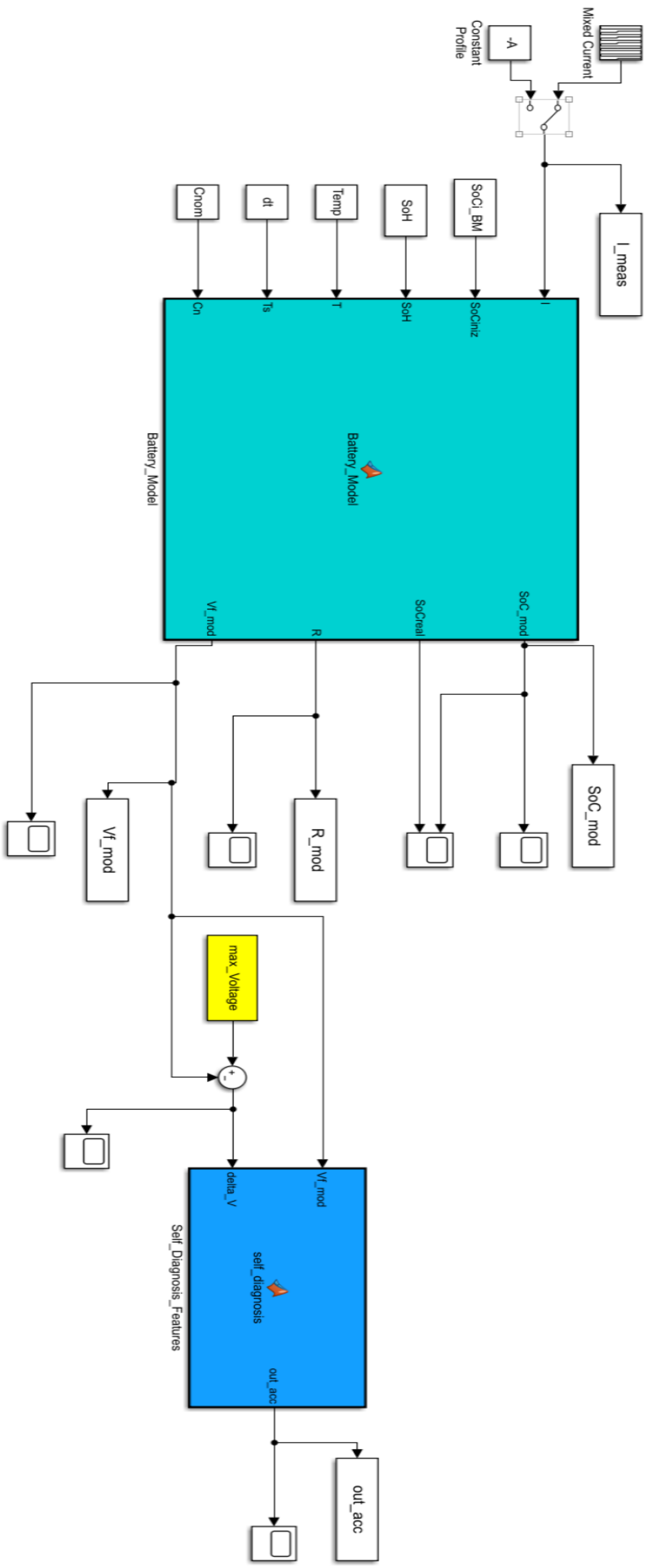


Figure 4.15 Simulink Model for the SDF of SoC Estimation

Now, we procced to perform the testing process on a new dataset from the same battery with the same current profile. This time, the SoCi for the new dataset is SoCi = [95%,75%,55%,35%,15%]. The following figures show the output of the SDF compared with the actual accuracy of the feed forward neural network obtained from the simulation environment. An explanation for how to read those figures can be found in Appendix A. Also, in Figure 4.21, we can see the clear performance of the SDF.

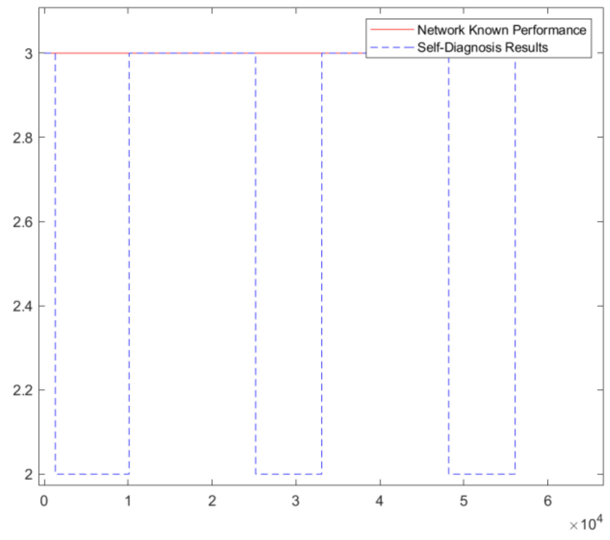


Figure 4.16 SDF Output for SoCi at 95%

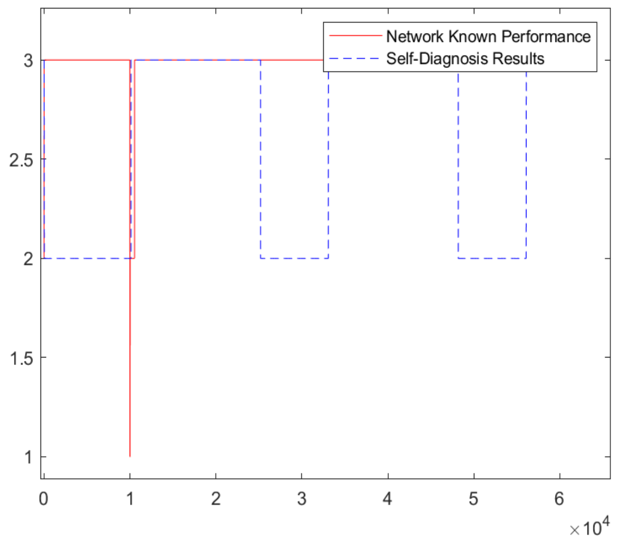


Figure 4.17 SDF Output for SoCi at 75%

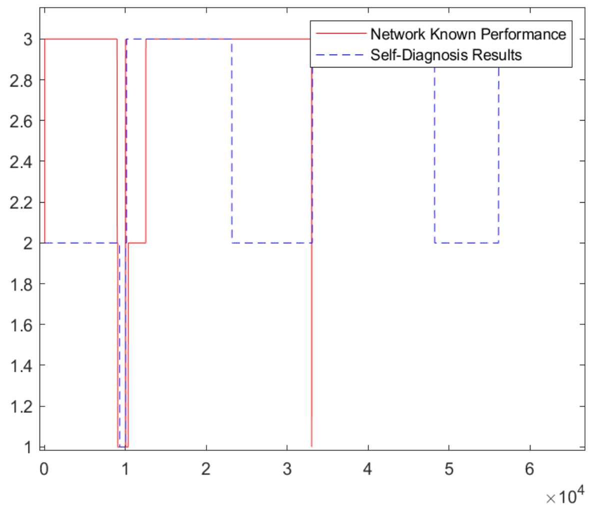


Figure 4.18 SDF Output for SoCi at 55%

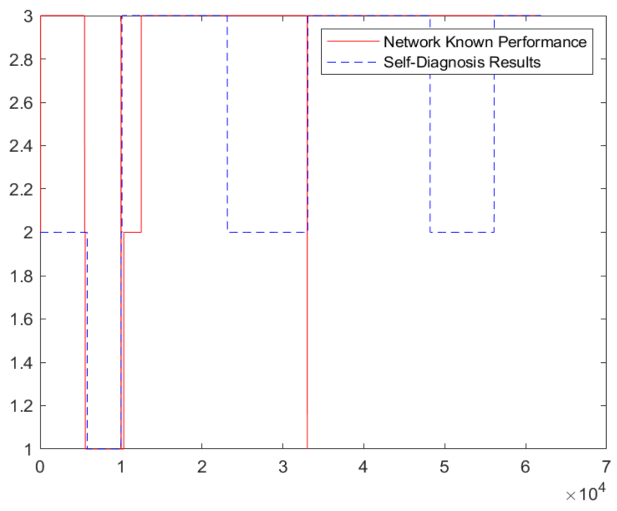


Figure 4.19 SDF Output for SoCi at 35%

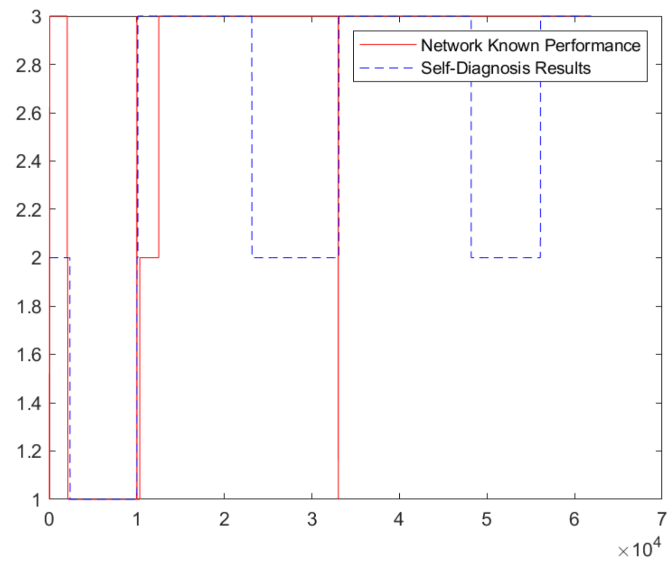


Figure 4.20 SDF Output for SoCi at 15%

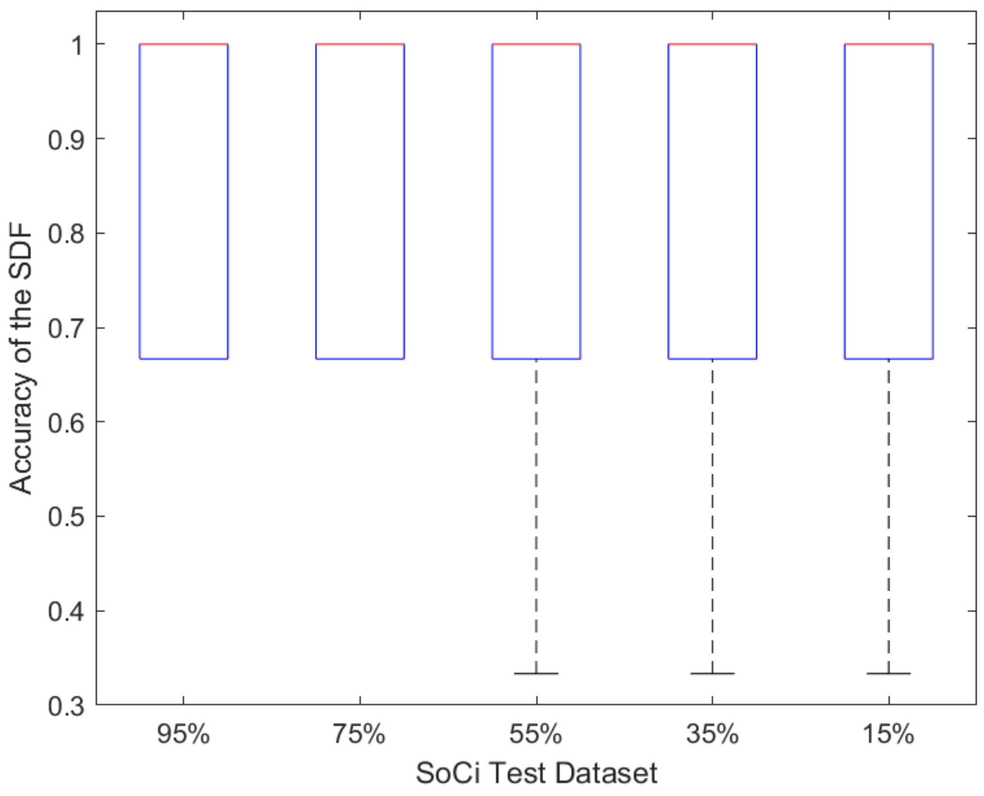


Figure 4.21 Performance of the SDF on the different dataset of SoCi

4.6 Discussion and Results

From the above figures, it is visible that in most cases for the different SoCi, the SDF is able to provide the correct assessment for the given estimation by the FFNN. Specifically, when looking at the boxplot at Figure 4.21, the accuracy of the SDF always has a median value of 100% accuracy, which means the SDF is working perfectly on self-assessing the performance of the network.

But it must be noted that this was only performed in a simulated environment and not on real data, which could yield different results. Another thing is that the tests that were performed are done on the same battery with the same current profile and with the same SoH (at 100%), leading to the fact that changing the current profile or SoH might cause the performance of the SDF to decline. This problem can be tackled on future works and more development of the methodology by building the SDF around different current profiles with varying the SoH.

Chapter 5

5 Self-Diagnosis Features for SoH Estimation of CNC Machine

The second application of the Self-Diagnosis Features is presented in this chapter. This time, it is applied into the process of *Estimation of the SoH for a CNC Machine*.

The following chapter gives a brief description about the CNC machine. Then, provides the estimation process of the SoH for a CNC machine using FFNN and the implementation of the SDF. Lastly, The result of the developed SDF is introduced.

5.1 CNC Machine

CNC (Computer numerical control) Machines can be defined as high-precision machines that regulate material elimination manufacturing operations that often require computerized control action to ensure high precision and efficiency. Elimination A manufacturing technique removes layers of material from a stock item known as workpiece using machine tools, resulting in a custom-made part. This sort of processing is unaffected by the workpiece's material composition: plastics, metals, foam, glass, and so on. Therefore, CNC machines are used in almost every aspect of industrial processing. Those machines are usually a SCARA or cartesian robotic configuration with the end-effector as a cutter [22].

Figure 5.1 illustrates the schematics of the CNC Machine.

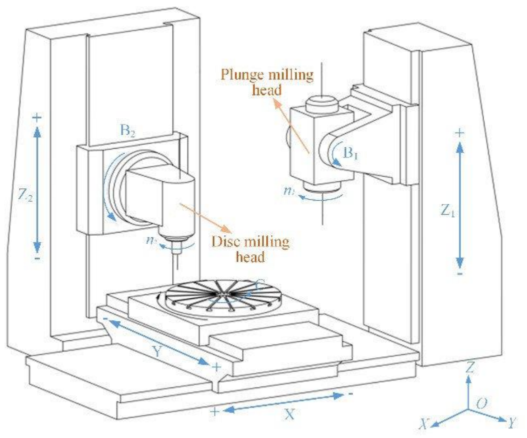


Figure 5.1 Schematics of A CNC Machine

5.2 Wear Estimation

The modeling for the cutter is very challenging since it depends on many variables, which includes the robotic configuration, environmental parameters and the state of health of the machine (also known as the wear condition). Those variables must be kept in check to assure the precision and efficiency of the CNC machine. But there is no known methods to evaluate the state of health of the end-effector. Although it is possible to check the temperature, voltage and pressure using sensors to estimate the SoH, it is very challenging to extract information related to the actual machine conditions. However, there exist some parameters that are straightly related to the SoH that can aid in the process of the estimating, the most notable are:

- Temperature;
- Friction Coefficient;
- Chip load.

Following previous researches done on the estimation of the SoH for the machine, it was found that the most effective parameter was the **friction coefficient**, since its major role for the interaction between the end-effector tool and the workpiece.

Knowing this, in our case study and application of the SDF, the friction coefficient (referred to as β) is **employed as the base for the estimation of the SoH**.

5.3 CNC Machine Model

The CNC machine consists of two parts. The first is the **mechanical part**, which is a simple milling machine that drives a rotational disc that moves in the piece direction for the cutting process. The second is the electrical part that is considered as a DC motor to drive the mechanical part. Figure 5.2 and Figure 5.3 visualize the mechanical and electrical parts respectively. An exhaustive explanation on how these parts work and their corresponding equations can be found in [22], since the modeling of the machine is not the main objective of this thesis work.

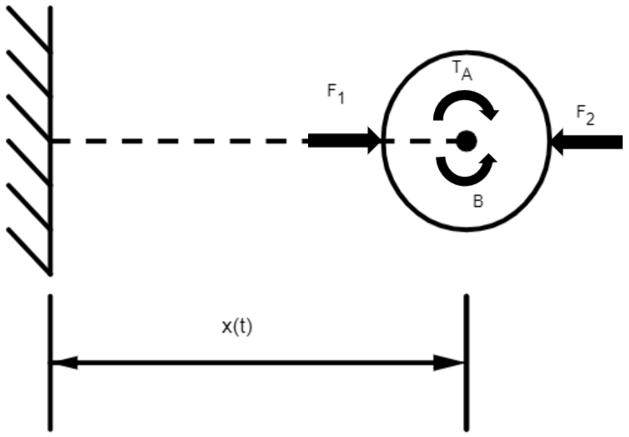


Figure 5.2 Mechanical Part

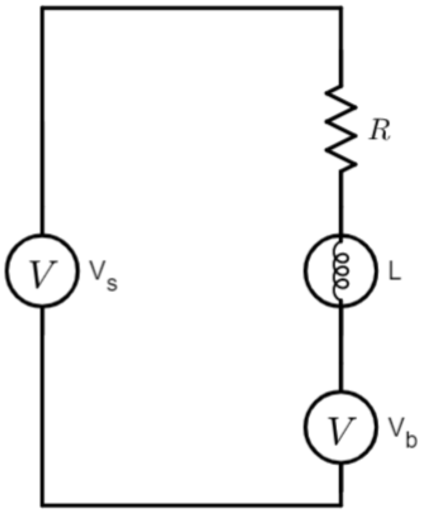


Figure 5.3 Electrical Part

Some of The significant quantities that can be found in the mechanical part are:

- Rotational Velocity;
- Linear Velocity;
- Horizontal Force;
- Dc motor torque;

- Constant Torque;
- Inertia of the motor.

And for the Electrical part:

- Supply voltage;
- Current;
- Inductance;
- Motor toque.

Similarly to the previous case study, This study is done in a simulation environment for the unavailability of a real data. The Simulink structure of the CNC machine that merges both the mechanical and electrical part can be seen in Figure 5.4. It was provided by Brain Technologies and was implemented in the MorePRO project.

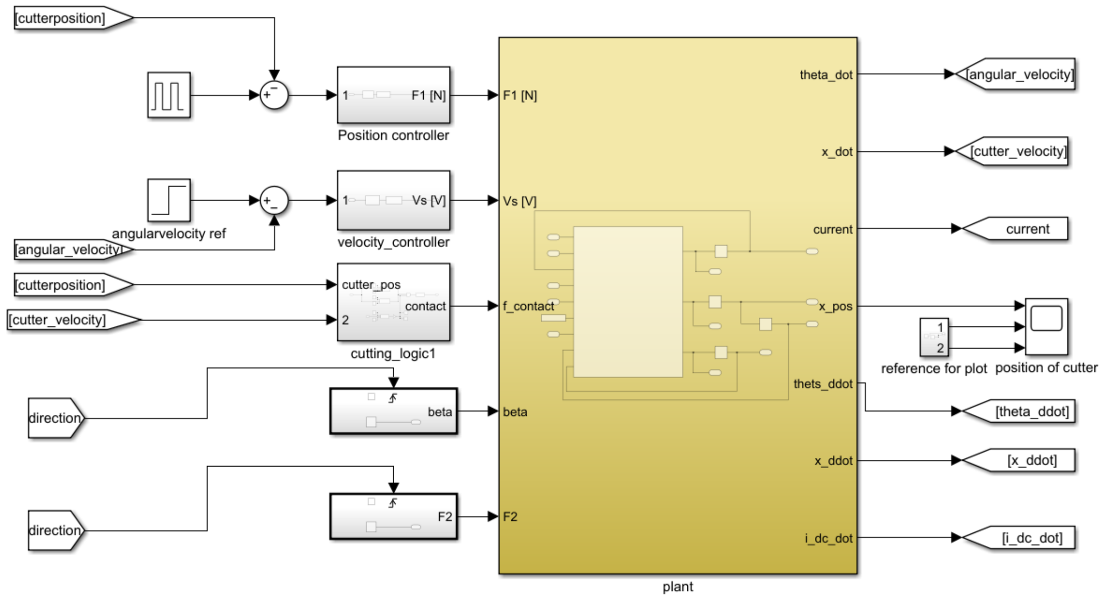


Figure 5.4 Simulink Structure of CNC Machine

Also, on the following figure, Another Simulink structure that represents the dynamical model existing within the plant shown above in Figure 5.4. More information about the dynamical model can be found in [22].

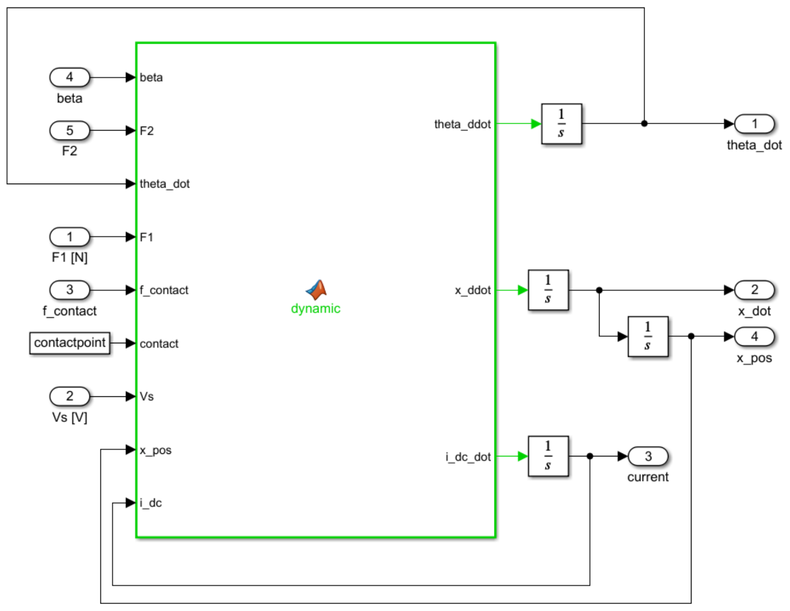


Figure 5.5 Dynamical Model

5.4 Estimation of the SoH using a FFNN

The estimation of the SoH is performed in a simulated environment. Leading to performing the data collection process in Simulink environment using the structure shown in Figure 5.4. Three simulated sensors are placed to retrieve four variables from the model: The angular velocity, the DC motor current and the friction coefficient *β*. And as noted before, the friction coefficient is used to extrapolate the SoH.

5.4.1 Training and Testing Dataset

The nominal working conditions of the machine that is used to obtain the training/testing dataset is shown in Table 2.

Parameter	Working value
Angular Velocity [rad/s^2]	210
Position reference [m]	0.45
Duty Cycle [%]	90
Number of cycles	100
Simulation time [s]	3600

Table 2 CNC Machine Nominal Condition

For the training dataset, the value of the friction coefficient is chosen to be ranging between 0.1 to 1 with a random interval. The reason for this is that when dealing with the estimation of SoH, the value of the parameter that is used to obtain the SoH (friction coefficient in this case) always stays the same for the duration of the simulation, since it takes a long time for the SoH to drop in value. This leads to the fact that if the neural network is only trained on one value of the friction coefficient, it will only be able to estimate that value when it is tested on different datasets, hence the usage of the ranging values of the friction coefficient.

The figure below shows the chosen friction coefficient profile to be estimated later by the neural network.

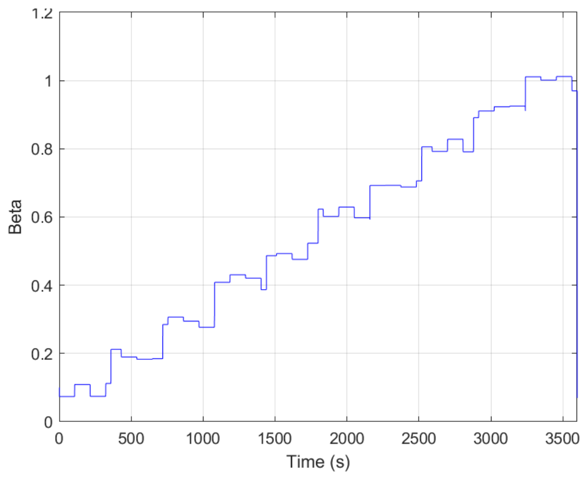


Figure 5.6 Friction Coefficient Training Profile

On the other hand, the testing datasets shall have only one value of the friction coefficient for each of them. Where the value of the friction coefficients for the different datasets is β=[1, 0.75, 0.5, 0.25].

5.4.2 Specification for the SoH FFNN

Following the data pre-processing and dataset randomizing using the K-fold cross-validation as explained in 3.1.3, the next step is the development of the FFNN that shall be used for the purpose of the estimation of the SoC. The Inputs to the network are the current and voltage, while the SoC is the output.

The next process is to present the specification of the FFNN used for the estimation purposes. The inputs to the network are the DC Motor current and the angular velocity, and the output is the friction coefficient. The specifications are shown in Table 3. The choice for the hidden layer neurons is done using the same method shown in 4.3.2.

Specifications	FFNN
Number of inputs	2 (DC Motor Current, Angular Velocity)
Number of neurons in the hidden layer	18
Number of outputs	1 (Friction Coefficient)
Activation Function in hidden layers	Sigmoid
Optimizer	Levenberg-Marquardt
Loss function	RMSE

Table 3 FFNN Specification

5.4.3 Testing and Training the FFNN

The FFNN is trained on the dataset obtained from the coefficient profile shown in Figure 5.6.

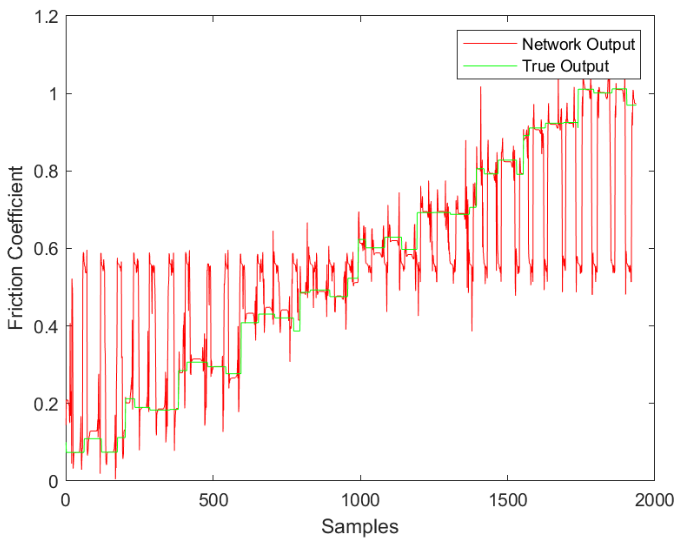


Figure 5.7 Network Estimation of Friction Coefficient

From Figure 5.7, it is noticeable that there is an expected change in the value of estimation whenever the value of the friction changes. This is caused by the change in the contact force. Whenever there is contact between the end-effector and the workpiece, a sudden change occurs for the DC motor current and the angular velocity. However, this does not affect the performance of the neural network as it is evident on Figure 5.8, as it shows the performance of the network on the different test dataset.

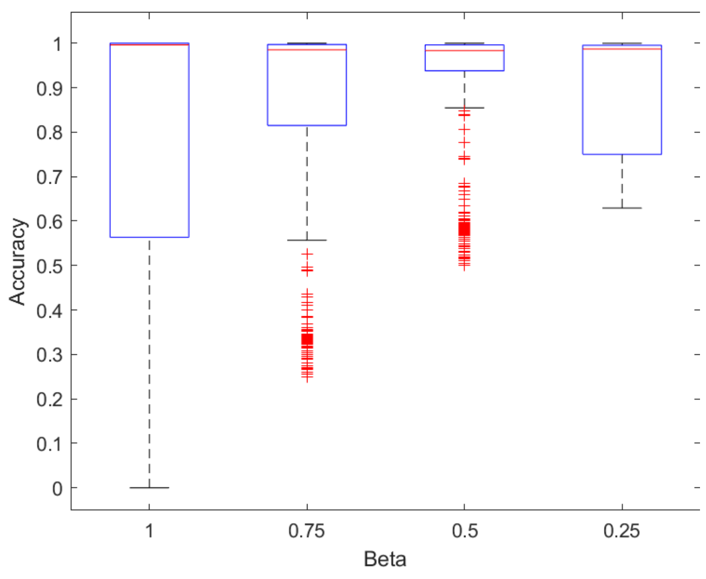


Figure 5.8 Performance of the FFNN output at different *β*

5.5 SDF Implementation for SoH of CNC Machine

Next, by knowing the performance of the FFNN, we procced to developing the SDF. And the first stage is to perform the data clustering using K-Means

5.5.1 Data Clustering for SoH of CNC Machine

The data clustering is performed on both dataset used for training and testing the neural network.

The clustering of the data is done using the DC motor current and the angular velocity.

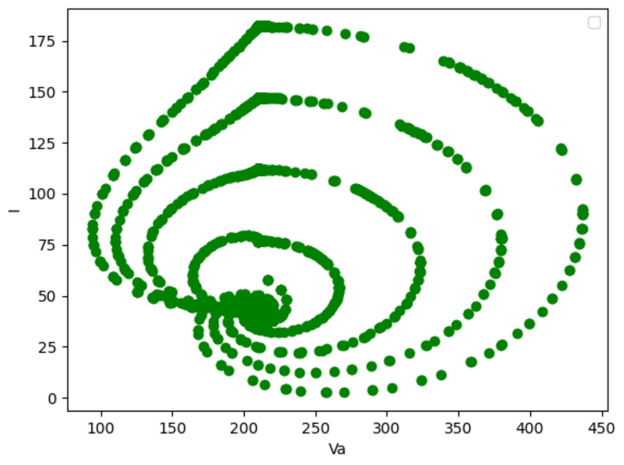


Figure 5.9 Unclustered Dataset of SoH

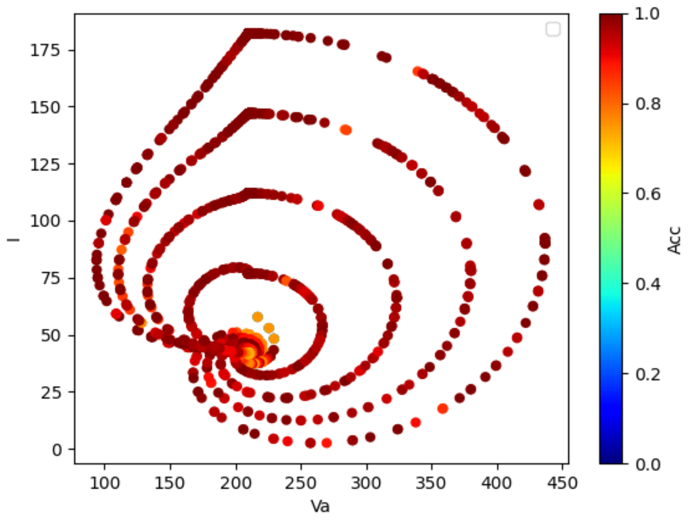


Figure 5.10 Unclustered Dataset with the known Accuracy of SoH Estimation

Figure 5.9 illustrates the uclustered dataset and Figure 5.10 represents the same dataset but with the known performance obtained by testing the network as shown in the boxplot at Figure 5.8.

As it is evident from the dataset in Figure 5.10, all of the samples have an accuracy above the 85% (red colored). This means there is no necessity for creating the performance levels using K-Means algorithm. we only need one performance level in this case, which is the high performance level. But this makes defeats the point of SDF if the FFNN is always able to estimate the correct value. Hence, we must force the FFNN to have a variation on its performance. This is done by performing the sensitivity analysis (discussed in 3.3.2). After the analysis it, the variable that is able to cause a noticeable change on the performance is chosen to create new testing dataset for the network and also it is considered to be the **effective clustering feature**.

5.5.2 Sensitivity Analysis for CNC Machine

Until now, the FFNN was tested using a dataset that provide a high performance. It was then decided to carry out with a sensitivity analysis that allows us to understand the model of the CNC Machine more in depth and which variables can allow for a variation or reduction on the performance of the network.

Many tests were done by varying the nominal working conditions shown in Table 2, but the results were also similar where the network was producing optimal results. The only choice left is to dive deep within the dynamical model of the machine (shown in Figure 5.5). And luckily after the many analyses on the different variables inside the dynamical model, two variables were found that cause a variation on the network performance, which they are:

- The Inertia of the motor and the cutter (I_n);
- The constant Torque (K_t)

5.5.2.1 Sensitivity Analysis for the Inertia

The first analysis is performed on the inertia of the motor and the cutter. The inertia depends on two parameters, the mass of the cutter and its radius. By varying these values, we can create different machines with different interias that can be similar to the real machines used in the production line, for the purpose of testing the neural network (shown in Table 4). It must be noted that the friction coefficient for all of the upcoming tests is considered to be $\beta = 0.5$ and that is because as shown in the boxplot at Figure 5.8, all of the coefficients used for testing yield similar performance results.

Machine	Mass (Kg)	Radius (m)	Inertia (Kg.m^2)
1	1.5	0.1	0.008
2	2	0.1	0.01
3	2.5	0.2	0.05
4 (Nominal Machine)	3	0.3	0.13
5	3.5	0.5	0.43
6	5	0.8	1.6
7	7	1	3.6

Table 4 CNC Machines with different Inertia

Now, the FFNN network is tested again on these different machines. The performance results can be in the following boxplot.

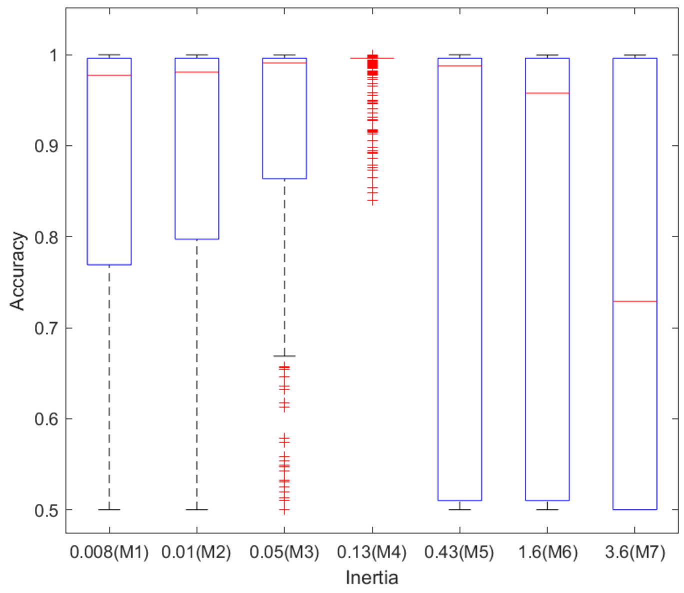


Figure 5.11 FFNN performance on the different Inertias

As seen in Figure 5.11, the network is able to perform with high accuracy on the different machines. This excludes the inertia as the choice of the effective clustering feature.

5.5.2.2 Sensitivity Analysis for the Constant Torque

The second analysis is performed using the motor constant torque. Similarly, to the previous analysis, different machines with different torques similar to real machines are chosen, with also the friction coefficient at 0.5.

Machine	Constant Torque
1	1
2	1.1
3	1.3
4 (Nominal Machine)	1.5
5	2
6	3
7	5

Table 5 CNC Machines with different Constant Torque

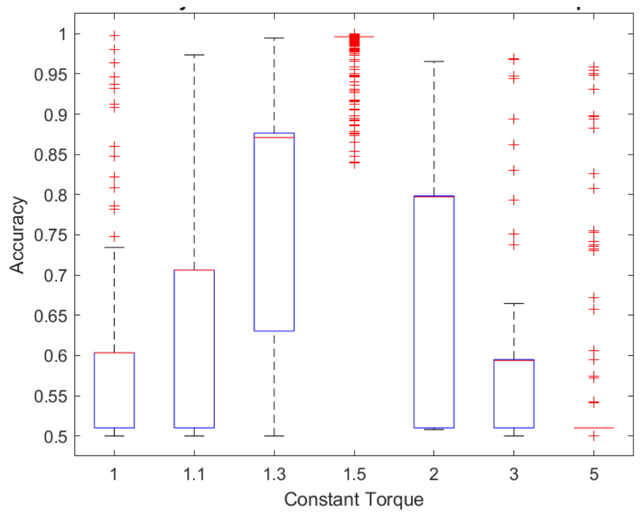


Figure 5.12 FFNN performance on the different Toques

It is evident from Figure 5.12 that the variation of the constant torque was able to produce different results for the FFNN, which is perfect for the implementation of the SDF. Meaning the constant torque is now chosen as the effective clustering feature that shall allow us to create different performance levels.

5.6 Data Clustering using the Effective Clustering Features

The constant torque is now added as a new variable to the clustering dataset. And this time, the K-Means clustering is performed using three variables: The angular velocity, DC motor current and the recently added constant Torque.

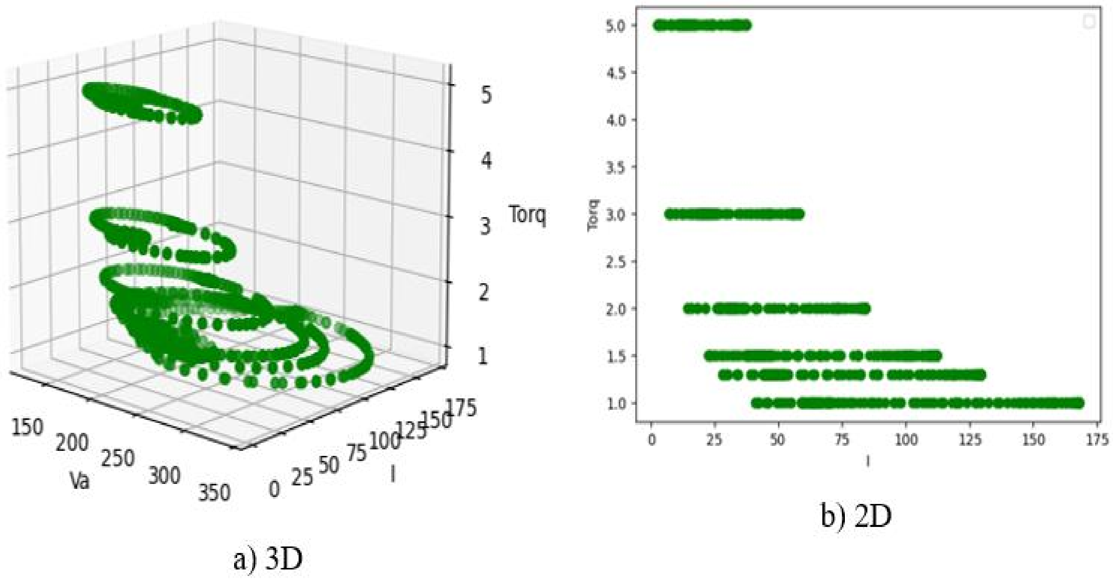
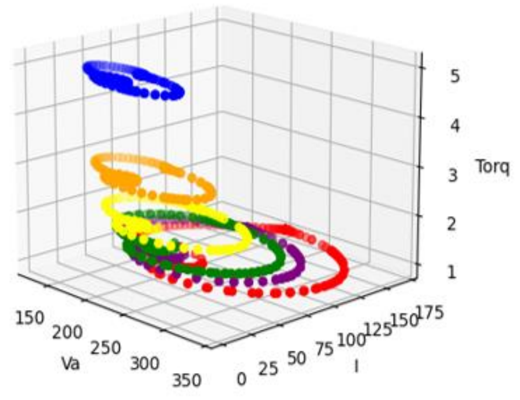
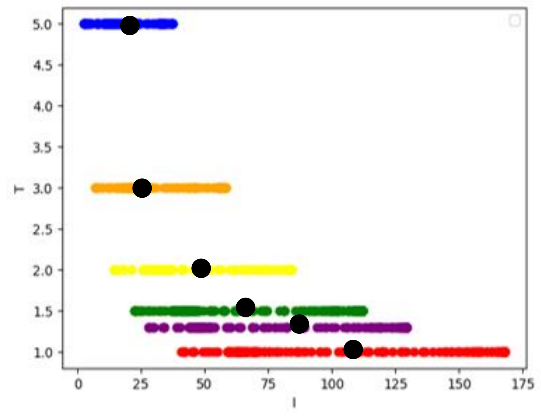


Figure 5.13 Unclustered SoH Dataset

The K-Means Results are illustrated in Figure 5.14.



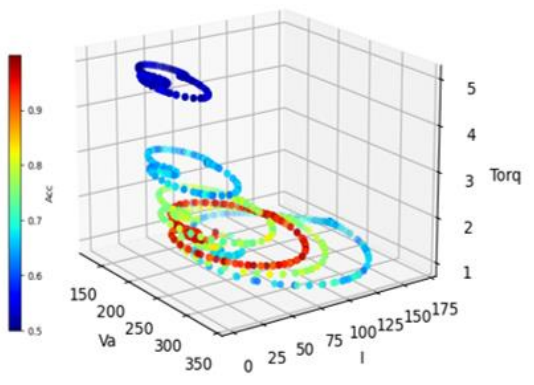
a) 3D



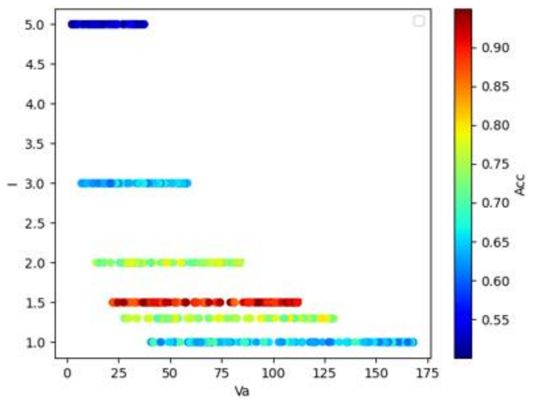
b) 2D

Figure 5.14 K-Means Clustering for SoH

As we can see in Figure 5.14, the number of clusters this time is 6 clusters. They were made this way to simplify the process of choosing the performance levels, since two clusters that share the same performance level can be in different places in the cluster space.



a) 3D



b) 2D

Figure 5.15 Dataset with the actual accuracy of the FFNN

Looking at Figure 5.15, we can confirm what was explained on the previous paragraph, where two clusters can share the same performance level while being far from each other.

Following the execution of the K-Means algorithm, it was found that the algorithm is able to find the correct match up with an accuracy of 90.7%.

5.6.1 Performance Levels and Centroids Extraction

The next step now is to define the performance levels. They are divides as follows:

- **High Performance Levels**: Includes all dataset that has an accuracy in the range of 100% to 81%, which includes one cluster only shown in Figure 5.16;
- **Medium Performance Level**: Includes all dataset that has an accuracy in the range of 80% to 65%, which includes two clusters shown in Figure 5.17;
- **Low Performance Level**: Includes all dataset that has an accuracy in the range of 64% to 0%, which includes three clusters shown in Figure 5.18.

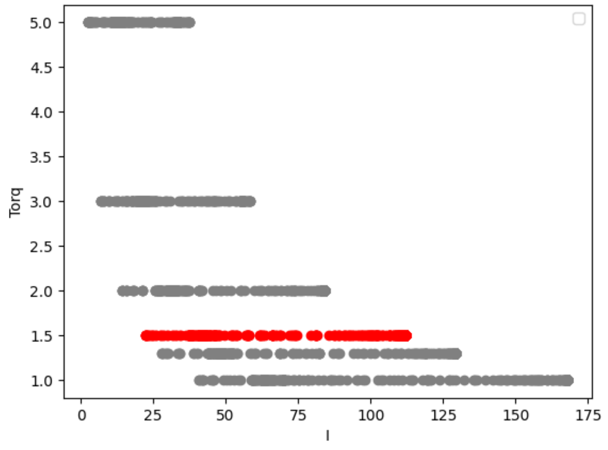


Figure 5.16 High Performance Level

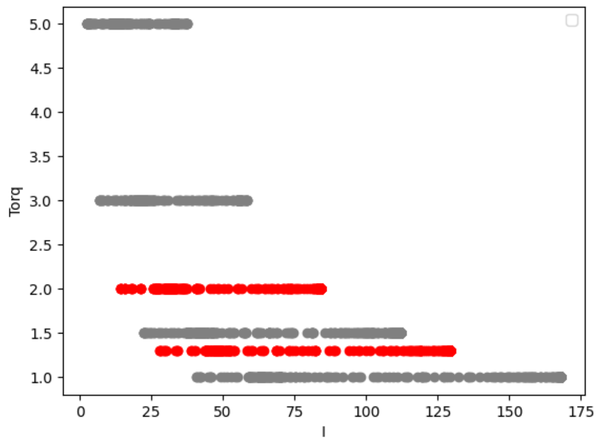


Figure 5.17 Medium Performance Level

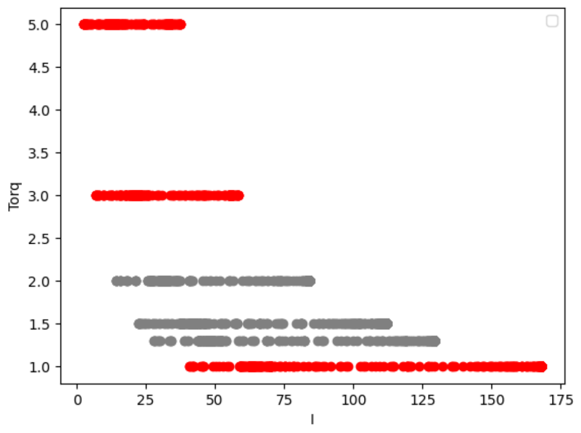


Figure 5.18 Low Performance Level

And now the last step for the clustering is the extraction of the centroids coordinates, as seen in Figure 5.19. Each coordinate vector is made out of three components: the constant torque, the angular velocity and the DC motor current respectively.

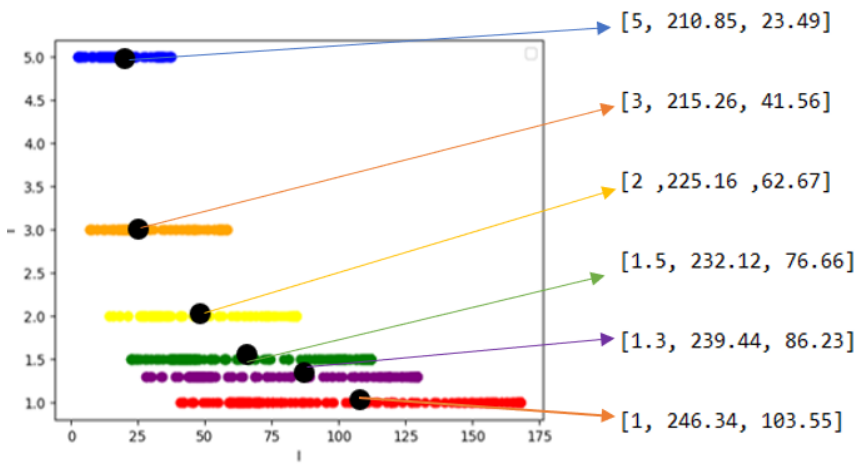


Figure 5.19 Centroids Extraction

5.7 Testing the SDF

Similarly, the previous case study, the next step is to test the SDF. In following graph, the Simulink structure for the CNC model with the SDF block is shown.

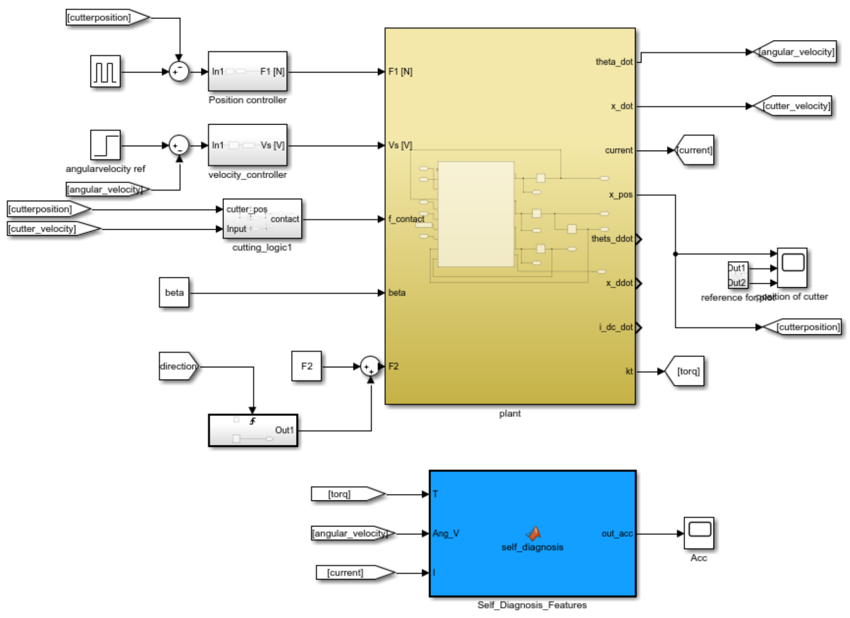


Figure 5.20 Simulink Structure with SDF for SoH

The testing process in done on different machines with the specifications shown on Table 6.

Machine	Constant Torque	Friction Coefficient
1	1.2	1
2	1.4	0.8
3	1.7	0.7
4	2.2	0.65
5	3.3	0.5
6	4.7	0.3

Table 6 CNC Machines for Testing

The results of the test are shown in the below plots with Figure 5.27 for showing the performance of the SDF on the different machines.

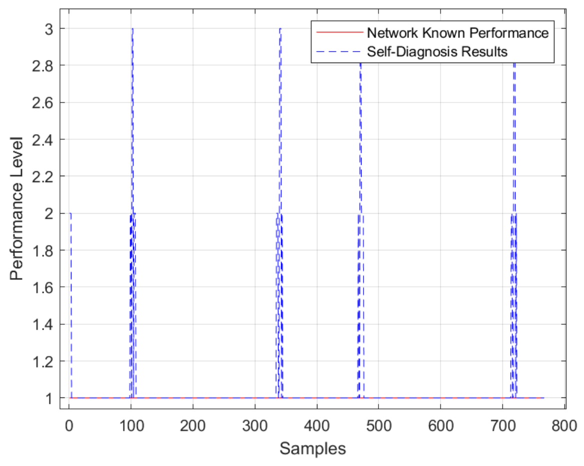


Figure 5.21 Machine 1 Test

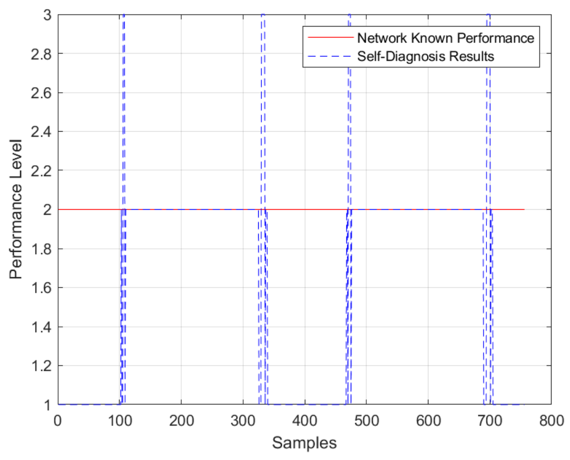


Figure 5.22 Machine 2 Test

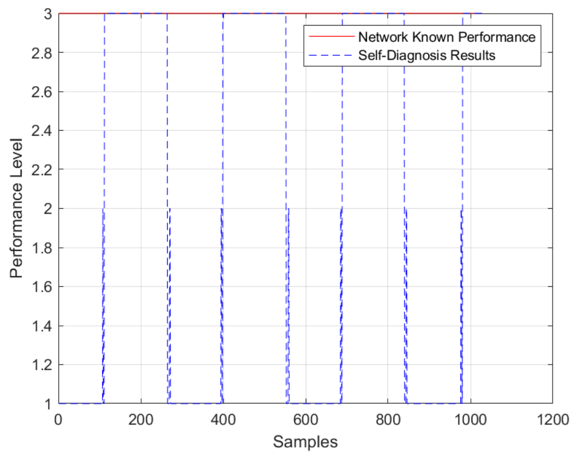


Figure 5.23 Machine 3 Test

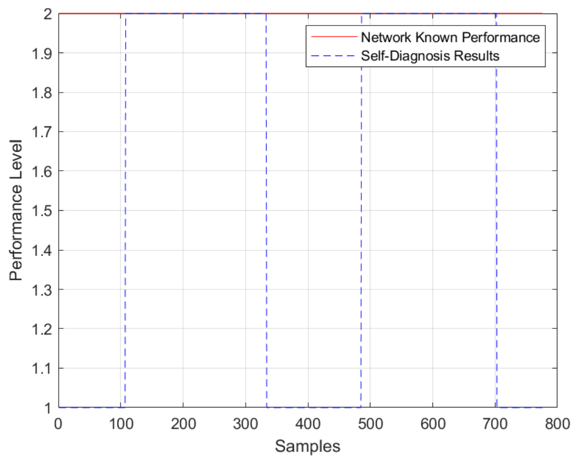


Figure 5.24 Machine 4 Test

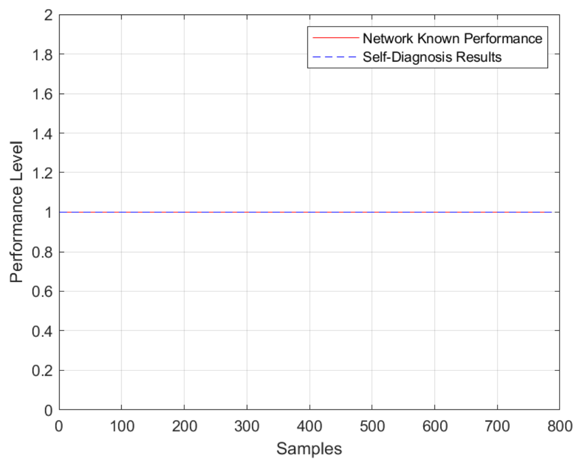


Figure 5.25 Machine 5 Test

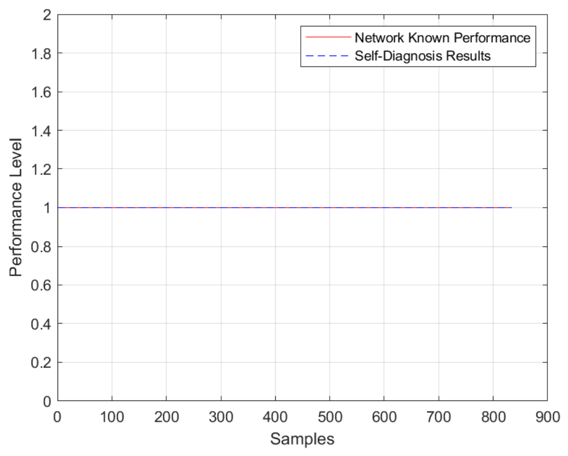


Figure 5.26 Machine 6 Test

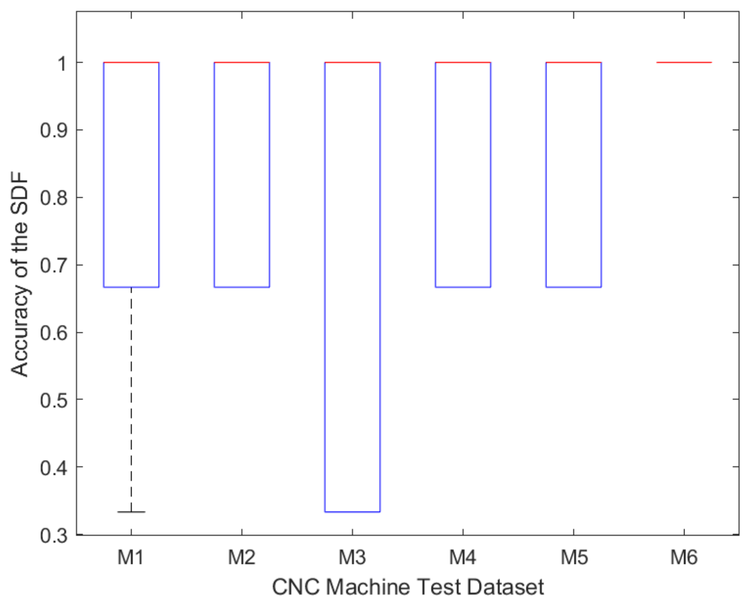


Figure 5.27 Performance of the SDF on the different Machines

5.8 Discussion and Results

Looking at the boxplot at Figure 5.27, the SDF is able to provide a very acceptable self-assessment of the accuracy for the FFNN used to estimate the SoH (the friction coefficient in this case). It is noticeable that the median value for most samples of the dataset lays at 100% accuracy, which is very acceptable.

However, these tests were only done on the case of the angular velocity being constant and not varying. In the case of its variation, the results of the SDF shall vary. This is going to be left for future works, since this is just the initial build of the SDF.

In addition to this, a notable discovery during this case study is that the use of neural networks for estimating the SoH of the CNC machine is a very optimal choice. This is because when we were in the estimation process, the network was able to estimate the SoH with a high accuracy, and we were forced to perform a sensitivity analysis to force it to have a lower performance.

Chapter 6

6 Conclusion and Future works

6.1 Conclusion

Lastly, it is possible to present all of the principal results achieved during the thesis work. Whose main objective was to develop new features that provide the machine learning algorithms the ability to self-assess their performance on-line. Following the path and methodology presented in the different chapters, the following conclusions are realized:

- First, the implementation of the new state-of-art technology of the self-diagnosis features that adopts the concept of data clustering has yielded very good results when it is equipped to a feedforward neural network. The SDF first application was the estimation of the SoC of a battery system, where the features shown very high accuracy for the self-assessment of the network for the estimation of the SoC at different levels of SoCi. The second application of the SDF was on the estimation of the SoH of the CNC machine, where it was also showing excellent results during the testing process, keeping in mind that the tests where done for a dataset with a varying constant torque and friction coefficient at the same time.

- Secondly, it has been demonstrated that the use of the deep neural network for the case of estimating the SoH of the CNC machine is very optimal. During the estimation process it was noticed that the network on the different considered values of the coefficient friction was able to estimate the value correctly. In previous case studies, the switching multimodal estimation technique was adopted, and now a new alternative is available which very easy to develop in contrast to the mentioned technique.

6.2 Future Works

The thesis has shown a promising future for the development of the self-diagnosis features in the field of the machine learning algorithms diagnosis.

In terms of the case studies of the work, it is evident that the data that was used is developed from a simulated environment of Simulink. The line action to be performed on the future works is to collect real data from the application plant in order to provide a validation for all the results obtained during the studies. This way, it’s feasible to verify the performance of the SDF on the different machine learning algorithms and improve its reliability.

In addition, the two performed case studies where only focused nominal plants without considering the possible effect of the change of the nominal conditions. In the future works, it is possible to considering a variation on the plant parameters to generalize the concept of the SDF for any possible setting.

7 Appendix A

7.1 Boxplots

The boxplot, also known as the exterior and quartiles diagram, is a visual representation used in statistics to illustrate the distribution of a sample using dispersion and position indices, as shown in Figure 7.1.

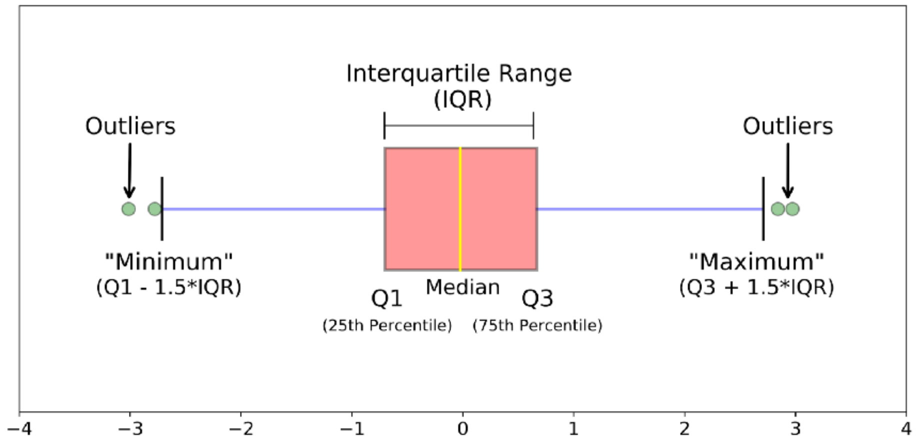


Figure 7.1 Boxplot

Figure 7.1 depicts a boxplot, which is represented by a rectangle divided into two pieces from which two segments emerge. The first and third quartiles, q1/4 and q3/4, respectively, define the box, which is split by the median, q1/2. The minimum and maximum values, also known as outliers, separate the segments.

7.2 Self-Diagnosis Features Output Plot

The plot obtained after performing the self-assessment test for the machine learning algorithms is used to show what performance level does each test sample belong to on the y-axis. It is divided into three values:

- The value "1" which correspond to the low performance level;
- The value "2" which correspond to the Medium performance level;
- The value "3" which correspond to the High performance level;

The output includes both the known performance level obtained from testing the machince learning algorithm in the simulated environment and the self-diagnosis features performance levels.

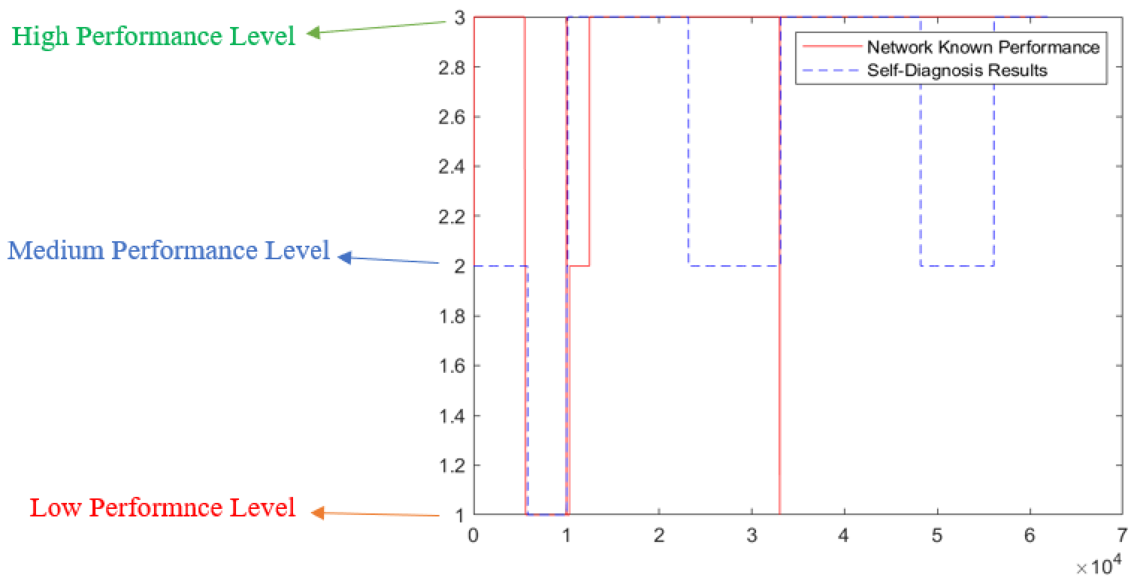


Figure 7.2 Self-Diagnosis Features Output

8 Bibliography

[1] H. Tillema, International Encyclopedia of Education (Third Edition), 2010, pp. 563-571.

[2] PoonPak-Lok, "A study on input domain partitioning," p. 2, 2002.

[3] M. Ryan, "In AI We Trust: Ethics, Artificial Intelligence, and Reliability.," 2020.

[4] P. Ongsulee, "Artificial Intelligence, Machine Learning and," Faculty of Science, Siam University, 2017.

[5] "Types of Machine Learning," [Online]. Available: https://www.javatpoint.com/types-of-machine-learning.

[6] J. Brownlee, "What is Deep Learning?," 2019.

[7] A. J. London, "Artificial Intelligence and Black-Box Medical Decisions: Accuracy versus Explainability," 2019.

[8] C. Molnar, Interpretable Machine Learning. A Guide for Making Black Box Models Explainable, 2022, p. Chapter 3.

[9] "Feedforward neural networks 1. What is a feedforward neural network?".

[10] A. Edgell, "Feedforward Neural Networks," 2021.

[11] A. Gupta, "A Comprehensive Guide on Deep Learning Optimizers," 2021.

[12] Seb, "An Introduction to Neural Network Loss Functions".

[13] T. Wood, "What is the Sigmoid Function?," 2018.

[14] "What is the Levenberg–Marquardt Algorithm?," [Online]. Available: https://www.statisticshowto.com/levenberg-marquardt-algorithm/.

[15] M. McGregor, "Clustering Algorithms in Machine Learning that All Data Scientists Should Know," 2020.

[16] I. Dabbura, "K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks," 2018.

[17] M. Rajaratne, "Data Pre Processing Techniques You Should Know," 2018.

[18] "Data Preprocessing," 2021. [Online]. Available: https://www.techopedia.com/definition/14650/data-preprocessing.

[19] "Feature Extraction," [Online]. Available: https://deepai.org/machine-learning-glossary-and-terms/feature-extraction.

[20] D. Faverato, "Virtual Sensing for the Estimation of the," Politecnico di Torino, Torino, 2020.

[21] K. Brush, "state of charge (SOC)," 2018.

[22] D. Zanon, "End-Effector Tools for State of Health Estimation: A Data Driven Approach," Politecnico di Torino, Torino, 2021.