Mittuniversitetet
MID SWEDEN UNIVERSITY

Predictive Maintenance in Smart Agriculture Using Machine Learning

A Novel Algorithm for Drift Fault Detection in Hydroponic Sensors

Ayad Shaif

Master Thesis
The main field of study: Computer Engineering
Credits: 30 hp
Semester, Year: Spring, 2021
Supervisor: Dr. Stefan Forsström
Examiner: Prof. Mikael Gidlund
Course code: DT005A
Degree program: Master of Science in Engineering – Computer Engineering

Abstract

The success of Internet of Things solutions allowed the establishment of new applications such as smart hydroponic agriculture. One typical problem in such an application is the rapid degradation of the deployed sensors. Traditionally, this problem is resolved by frequent manual maintenance, which is considered to be ineffective and may harm the crops in the long run. The main purpose of this thesis was to propose a machine learning approach for automating the detection of sensor fault drifts. In addition, the solution's operability was investigated in a cloud computing environment in terms of the response time. This thesis proposes a detection algorithm that utilizes RNN in predicting sensor drifts from time-series data streams. The detection algorithm was later named; *Predictive Sliding Detection Window* (PSDW) and consisted of both forecasting and classification models. Three different RNN algorithms, i.e., LSTM, CNN-LSTM, and GRU, were designed to predict sensor drifts using forecasting and classification techniques. The algorithms were compared against each other in terms of relevant accuracy metrics for forecasting and classification. The operability of the solution was investigated by developing a web server that hosted the PSDW algorithm on an AWS computing instance. The resulting forecasting and classification algorithms were able to make reasonably accurate predictions for this particular scenario. More specifically, the forecasting algorithms acquired relatively low RMSE values as ~0.6, while the classification algorithms obtained an average F1-score and accuracy of ~80% but with a high standard deviation. However, the response time was ~5700% slower during the simulation of the HTTP requests. The obtained results suggest the need for future investigations to improve the accuracy of the models and experiment with other computing paradigms for more reliable deployments.

Keywords: Predictive Maintenance, Internet of Things, Machine Learning, RNN, Forecasting, Classification, LSTM, CNN-LSTM, GRU, Sensor Drift Fault

Acknowledgments

The accomplishment of this thesis could not have been possible without the guidance and feedback of Dr. Stefan Forsström, my supportive academic supervisor.

I would like to express my gratitude to Luca Beltramelli, Aamir Mahmood, Elijs Dima, and Johannes Lindén, from the Department of Information Systems and Technology at Mid Sweden University, for the endless support and discussions.

In addition, I would humbly like to extend my appreciation to my examiner, prof—Mikael Gidlund, for his sincere guidance and encouragement throughout this thesis.

A special thanks to my supervisor, Peter van der Meulen from Knightec, for his assistance in forming the thesis proposal and milestones.

Last but not least, I would like to express my most profound appreciation to my family for their unconditional love and support.

Table of Contents

Terminology

Abbreviations

ANOVA	Analysis of Variance
AWS	Amazon Web Services
CI	Confidence Interval
CNN	Convolutional Neural Network
CRISP-DM	Cross-Industry Process for Data Mining
GRU	Gated Recurrent Unit
HTTP	Hypertext Transfer Protocol
IoT	Internet-of-Things
JSON	JavaScript Object Notation
LSTM	Long Short-Term Memory
ML	Machine Learning
NN	Neural Network
PSDW	Predictive Sliding Detection Window
RMSE	Root Mean Square Error
RNN	Recurrent neural network
SD	Standard Deviation
Tukey's HSD Test	Tukey’s Honestly Significant Difference Test

1 Introduction

The internet of things (IoT) is a communication paradigm that extends the internet by a balanced fusion between the physical and the digital world by sensors and actuators. The utilization of IoT technologies has been increasing rapidly in the past decades, making it one of the most revolutionizing advances in modern history. This technological advancement allowed the establishment of new "smart" applications that benefits from the sensor–actuator interplay. One of the main benefits of IoT is gathering data better to understand the assets and liabilities of the application. This could be achieved by extracting insights from the collected data and then interpret them for further improvements or research. [1]

In this regard, one prominent area that was enabled with the help of IoT advancements in predictive maintenance. The main idea of this kind of maintenance technique is that they use the gathered data from the sensing devices to estimate the condition of the assets. Predictive maintenance techniques play an essential role in maximizing the assets' lifetime by avoiding failures and any performance issues. [2]

One particular use case that uses IoT and predictive maintenance technologies is in smart hydroponic agriculture. Here, IoT devices such as sensors and actuators are utilized to control the farming environment to preserve the health of the crops. For instance, hydroponic sensors can detect and regulate the hydroponic environment's chemical and physical properties (e.g., Temperature, Humidity, pH level, etc.). [3] In such scenarios, predictive maintenance is used to effectively maintain the functionality of the deployed IoT devices and help detect any anomalies that could potentially affect the health of the crops. [4]

1.1 Background and Problem Motivation

In general, smart agriculture’s success depends on the inter-functionality between IoT devices deployed on a flexible and scalable infrastructure such as cloud computing services. [5][6] A typical smart agriculture setup consists of IoT devices such as sensors, actuators, and embedded systems that monitor the crop status and environment. This means that any occurring faults in those IoT devices might expose the whole system to the risk of failure and affect the quality of the crops. [7]

In smart agriculture that utilizes hydroponic techniques, the sensors' sensitivity is often affected by environmental contamination that accelerates the degradation of the deployed sensor. This usually causes error readings or even hardware failure that requires maintenance. This higher maintenance rate often leads to significant expensed to the business in terms of time and cost.

The sensors in smart hydroponic agriculture are traditionally maintained with reactive or preventive techniques. These techniques are considered ineffective and unsustainable, especially when deployed in larger areas where the sensors have a high recurrence rate of failures. In reactive maintenance, sensor repairs occur after the failure is detected, which may damage the crops. On the other hand, preventive maintenance tends to establish planned schedules for maintaining the hydroponic sensors regardless of the sensor's condition, which is considered inefficient and expensive in terms of time consumption and execution costs.

Therefore, the area of exploring cost-effective and reliable techniques for detecting failures in sensors from data streams has received growing attention. [8] Hence, modern maintenance schemes use data to predict the performance of the deployed sensors; there is a growing need to explore new methods that assist the sensor nodes with prediction mechanisms that increase the maintenance efficiency by automatically detecting sensor drifts. [9][10] There is also a growing demand in exploring different setups of machine learning algorithms that can be applied to different maintenance scenarios with better prediction performance. [8][11]

1.2 Overall Aim

This thesis investigates the use of machine learning algorithms to improve the performance of detecting failures in hydroponic sensor data streams. It is expected that automating the detection by introducing a window detection that enhances machine learning will improve maintenance efficiency. The aim of this thesis is therefore to investigate this problem area through the means of an implemented proof-of-concept. The implementation will be used to test and evaluate the failure prediction mechanisms in a scalable cloud environment. As the defined problem suggests, the proposed solution should be evaluated in terms of accuracy metrics and response time.

1.3 Scientific Goals

The scientific goal of this thesis is to propose an approach that uses machine learning to predict abnormal trends in sensor data streams. Therefore, this study will provide new insights into machine learning in predicting the maintenance of hydroponic sensors. Furthermore, this research also contributes to the field by exploring the efficiency of utilizing the proposed solution in a scalable environment. The developed solution will be deployed on a cloud computing platform and evaluated in time to respond to HTTP requests. More precisely, the concrete scientific goals of this thesis are to:

1- Propose an efficient novel algorithm for detecting sensor drift faults from time-series data streams using forecasting and classification recurrent neural networks.

2- Propose a strategy for utilizing supervised learning techniques in classifying sensor drift fault in data streams.

3- Review the performance of different recurrent neural networks in terms of accuracy for detecting sensor drift faults in this particular scenario.

4- Determine the tradeoffs of deploying the proposed novel algorithm in a cloud computing environment in terms of response time for this particular scenario.

From these goals, the expected main contributions of this thesis will be an algorithm that utilizes machine learning to detect sensor drift faults, a method for labeling sensor data streams for this particular scenario, a performance comparison of three prominent machine learning algorithms, and an evaluation of the developed algorithm in a cloud environment.

1.4 Thesis Milestones

The overall aim and scientific goals in this thesis can be achieved with the help of the five following thesis milestones:

1- Conduct a literature review to find the three most prominent solutions used in recurrent neural networks for time-series data streams.

2- Design the experimental setup for:

   a- Cloud computing environment for the IoT solution.

   b- Sensor drift fault prediction with machine learning.

3- Implement the designed components of the system for:

   a- IoT infrastructure for data generation and collection.

   b- Sensor drift fault prediction model deployed into the cloud environment.

4- Measure the performance of the implemented system in terms of:

   a- Average response time from the cloud computing environment.

   b- Accuracy metrics for the sensor drift fault prediction model.

5- Analyze the obtained results to evaluate the system's performance in terms of accuracy metrics and response time.

1.5 Scope

This thesis focuses mainly on proposing a predictive maintenance approach for IoT sensors used in smart agriculture applications. The thesis consists mainly of two parts for examining the utilization of machine learning algorithms in predicting sensor failures and the feasibility of implementing the proposed solution in a cloud platform. In the prediction algorithms, the thesis focuses on implementing common recurrent neural networks (i.e., LSTM, CNN-LSTM, and GRU) to forecast and classify failures caused by sensor wear-out (e.g., sensor drift faults). The developed algorithm is evaluated with standard performance metrics, namely RMSE for forecasting and F1-score and accuracy for classification. In cloud platform implementation, the thesis will examine the feasibility of implementing the developed solution by only focusing on the response time.

1.6 Outline

This thesis is organized as follows: Chapter 2 provides an overview of Internet-of-Things, Reliability theory, Cloud Computing, and Machine Learning. The chapter also presents some related works in the field of hardware failure detection and predictive maintenance. Chapter 3 presents the scientific methodology and the thesis milestones that were considered to fulfill this thesis's purposes. Chapter 4 provides a comprehensive comparison of two cloud computing platforms and ended with a motivation for the chosen platform. Chapter 5 presents the procedure of implementation and evaluation of the thesis setup. Chapter 6 demonstrates the obtained results. In chapter 7, the analysis of the obtained results is presented along with a relevant discussion on the ethical and social aspects in this thesis. Finally, a conclusion of the study is drawn in chapter 8.

2 Theory

This chapter describes both the most relevant theories to this thesis and presents some related research works. In the case of relevant theory, the chapter is divided into four main sections, namely *Internet-of-things*, *reliability theory*, *cloud computing*, and *machine learning*. The section is then finalized with a comprehensive presentation of the related research conducted in predictive maintenance and sensor fault detection for Internet-of-things use-cases.

2.1 Internet-of-Things

Internet-of-Things (IoT) is a paradigm that uses modern wireless communication to allow physical objects (e.g., mobile phones, sensors, and actuators) to collaborate in accomplishing the planned task for a particular application. The concept of IoT and its enabling technologies are expected to impact the future of society on different levels. On the personal level, IoT solutions will enhance its users' lifestyles by providing functionalities that benefit their day-to-day activities with applications such as assisted living and personal care. On the other hand, IoT can increase the value of the business at the enterprise level by optimizing its processes, logistics, and working routines. [1]

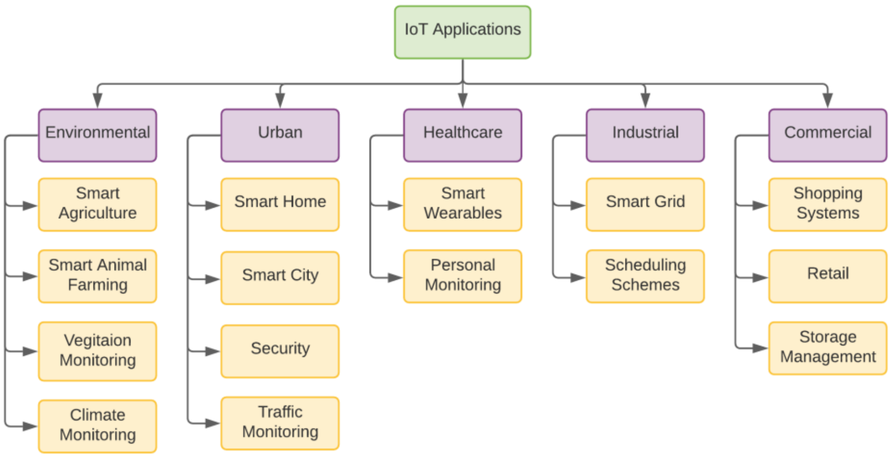


Figure 2.1, Typical IoT applications and use-cases.

Figure 2.1 demonstrates some of the main applications that make use of IoT technologies.

One example of a typical IoT application is in *hydroponic agriculture*. Here, the environment relies on eliminating soils in the agricultural procedure as the crops are nurtured using water and different minerals. The environments in hydroponic agriculture tend to be systematically controlled to maintain the optimum health of the crops. The control includes essential aspects such as nutrients, climate, and utilized energy, and water. In this kind of application, IoT provides functionalities that allow the facilities to save resources (e.g., energy consumption and water usage) and better maintain the health of the crops using different devices and sensors. [12][13][14]

On the other hand, the paradigm introduces several challenges that need to be addressed, especially when advanced features are introduced. Some examples of those challenges are security, privacy, resource efficiency as well as scalability. For instance, features that allow the devices to interoperate and scale-up freely could be prone to security attacks and higher maintenance costs. [15]

2.1.1 Visions of IoT

Three main visions collaboratively form the IoT paradigm. The first vision focuses on the network and communication aspect of the paradigm and will be referred to as *network-oriented* vision. The second vision involves the utilized physical objects that enable the deployment of the paradigm; this vision will be referred to as a *things-oriented* vision. The third and final vision concerns unique device addressing, data representation, and storage; this vision will be *semantic-oriented*. [1]

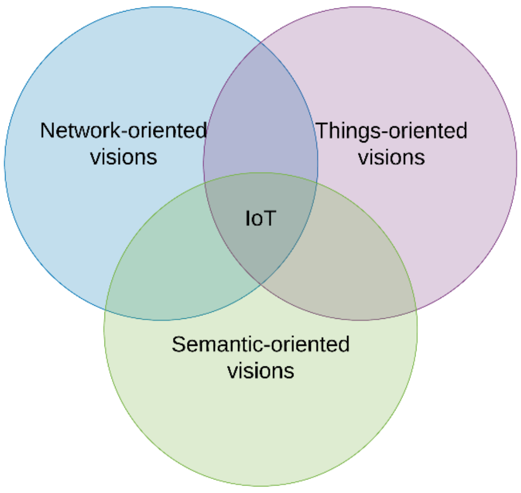


Figure 2.2, Three main identified visions for IoT.

The three visions with some of their main elements are presented in the Venn diagram demonstrated in Figure 2.2. As shown, the intersecting point where all three visions overlap is the IoT paradigm, as it requires all visions to collaborate and coordinate at some level.

2.1.2 Elements of IoT

As suggested by [16] most IoT solutions consist of a group of elements that jointly collaborate to achieve the desired application results.

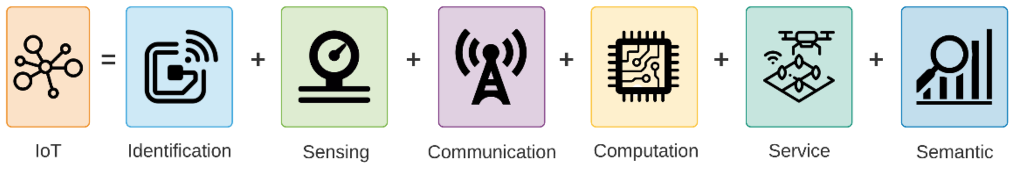


Figure 2.3, The six main elements that build IoT solutions.

Figure 2.3 demonstrates the six most common elements that enable IoT solutions to their fullest potential. Those elements can be framed as identification-, sensing-, communication-, computation-, service-, and semantics-element.

The *identification element* refers to the ability to identify each connected physical device uniquely. This could be achieved with the help of different naming and addressing methods such as Electronic Product Codes (EPC) and ubiquitous codes (uCode).

In the *sensing element,* all desired data are gathered from the connected IoT devices (e.g., sensors) and transferred further to be stored into a database or processed on the cloud. The sensing element is often linked to a Single Board Computer (SBC) to maintain its connection with a central unit. Once a sufficient amount of data is gathered from the IoT devices, many smart functionalities can be acquired through data analysis and interpretation.

The *communication element* consists of two main components, namely technologies (e.g., Near Field Communication (NFC) and Radio-frequency identification (RFID)) and protocols (e.g., Bluetooth and Wi-Fi). Those components allow IoT devices to communicate with each other easily. This element enables IoT devices to communicate in a heterogeneous and noisy environment.

The *computation element* is considered to be the brain of a typical IoT solution. In this element, different processing devices such as microcontrollers (e.g., Arduino) and microprocessors (e.g., Raspberry Pi) operate solely or jointly to run various applications. However, many platforms and systems are designed to provide IoT functionalities, such as Operating Systems and Cloud platforms.

The *service element* contains four main categories of services that connect real-world objects to the virtual world. Those categories are Identity-related -, Information Aggregation-, Collaborative Aware- and Ubiquitous-Services. The identity-related services manage the identification of physical objects and keep track of the registered ones. The Information Aggregation Services collect and analyze raw sensory data to satisfy the main IoT application. The Collaborative-Aware Services make use of the collected raw data to make decisions and interpretations. Ubiquitous Services, on the other hand, concerns the availability of different collaborative-aware services.

Finally, the *semantic element* focuses on the process of extracting knowledge from the provided services. This implies that the collected data will be discovered and modeled to analyze and acquire the most relevant information.

2.2 Reliability Theory

This section describes the main concepts and theories in reliability and failure analysis. The section starts with a general description of reliability, followed by a broad definition of failure and its main modes and metrics.

2.2.1 Reliability

Reliability [17] can be defined as the ability of an item (product or system) to maintain its functionality at the end of a particular mission. The concept of reliability can also be defined differently based on interest, such as in Engineering and Statistics.

In the context of engineering, reliability can be defined as the practice that deals with designing and analyzing the items to extend their lifetime. Here, the goal is to minimize the factors contributing to failures and therefore lead to item wear-out by avoiding harmful environmental deployments or providing preventive maintenance.

On the other hand, the probabilistic view quantifies the reliability by developing mathematical equations representing reliability as a probability function.

$$P(T \leq t) = \int_0^t f(\theta)\, d\theta = F(t), \qquad t \geq 0 \tag{2.1}$$

In Equation 2.1, the probability function describes the unreliability of the item in achieving the desired functionality. Here, the notation T denote to time-to-failure of an item, and it can also be represented by the probability density function (pdf) up to the end time of the mission t, $f(t)$.

$$R(t) = 1 - F(t) = \int_t^\infty f(\tau)\, d\tau \tag{2.2}$$

In contrast, the survivorship of the item in performing the given task can be described by Equation 2.2.

2.2.2 Failure

Failure [18] is the phenomenon that occurs when an item stops operating correctly. An essential parameter regarding failure and reliability analysis of items and systems is *failure-free time* (or failure-free operating time). This parameter is treated as a random variable and expresses the operability of the item without the interference of failures. Typically, the failure-free time of the items is long, where failures occur after a specific interval of time.

On some occasions, failures may occur way earlier on the item due to transient events at turn-on. The failure analysis is generally assumed that the items are in new condition or free of defects at the beginning of failure-free time (i.e., $t = 0$). There are different types of failures that can be categorized as mode-, cause-, effect-, and mechanism failures and can be defined as follows:

1- Mode: The mode refers to the observed local symptom of the item (e.g., short, drift, and functional fault for electronic components or brittle fracture, buckling, fatigue for mechanical parts).

2- Cause: The cause, on the other hand, refers to the causes that lead to the item's failure. Here, the causes are categorized as internal (e.g., item weakness/wear-out) or external (e.g., misuse or error in design, production, or use).

3- Effect: The effect which occurs as a result of failures. The impact of the failure can be classified based on the level of severity (e.g., non-relevant, minor, major, critical) or group of hierarchy (e.g., primary or secondary).

4- Mechanism: The nature of the occurring failure could be, for instance, physical, chemical, or process-based.

Another term that should be distinct is a fault. In the context of reliability theory, a fault is defined as the down state of an item that failures or defects could cause.

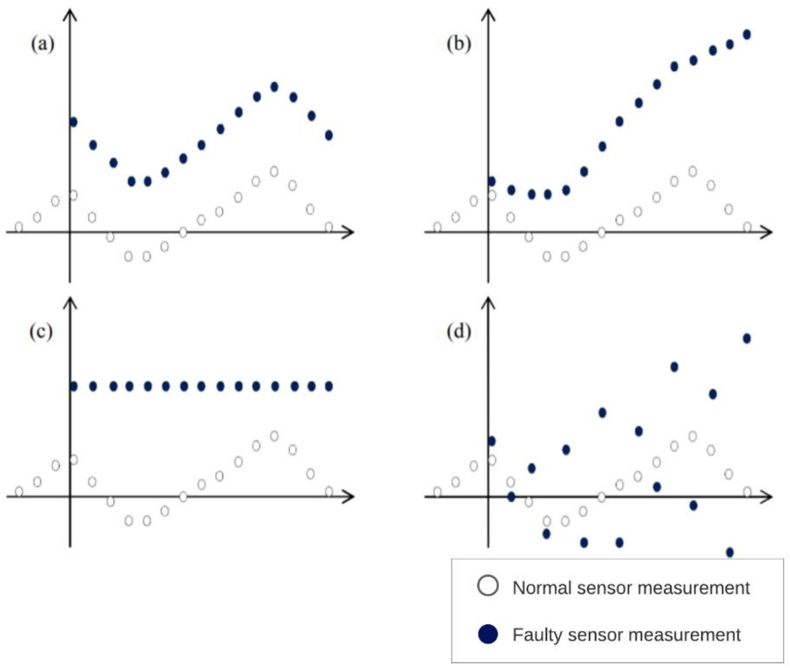


Figure 2.4, The four most common sensor failures: (a) bias, (b) drifting, (c) complete failure, and (d) precision degradation. [30]

For instance, in items like sensors, four common types of faults can highly affect their reliability. Figure 2.4 demonstrates the fault types and categorizes them as (a) bias, (b) drifting, (c) complete failure, and (d) precision degradation.

2.2.3 Failure Rate

Failure rate (also called hazard rate) describes the frequency of failures that occurs on the item per time unit. Failure rates are often denoted by the Greek letter λ. Failure rates are generally affected by the lifetime of the items, i.e., measures differently over time. For example, the failure rate of a brand-new vehicle is significantly lower than the failure rate of a five-year-old car, and thus more service and maintenance are required.

$$\lambda(t) = \frac{R(t_1) - R(t_2)}{(t_2 - t_1) \cdot R(t_1)} = \frac{R(t) - R(t + \Delta t)}{\Delta t \cdot R(t)} \qquad (2.3)$$

The failure rate at time points t and $\lambda(t)$ can also be defined as the probability for the occurrence of failure within a specified time interval, given that the item has survived until time t, see Equation 2.3. The equation consists of a failure distribution function $\lambda(t)$ and a reliability function $R(t)$ (i.e., probability that no failures have occurred until time t).

$$h(t) = \lim_{\Delta t \to 0} \frac{R(t) - R(t + \Delta t)}{\Delta t \cdot R(t)} \qquad (2.4)$$

In continuous intervals and smaller intervals, the failure rate can be treated as an instantaneous hazard function $h(t)$ as Δt approaches zero; the representation is presented in Equation 2.4.

2.2.4 Failure Modes and Characteristics

The failure modes are generally represented by a curve known as the bathtub curve. The curve is plotted with the failure rate, $h(t)$, in the y-axis and time t in the x-axis. The bathtub curve consists of three main phases: early failures, failures with constant failure rate, and wear-out failures. In that regard, each phase represents a specific mode of failure with characteristics that distinguish them from each other.

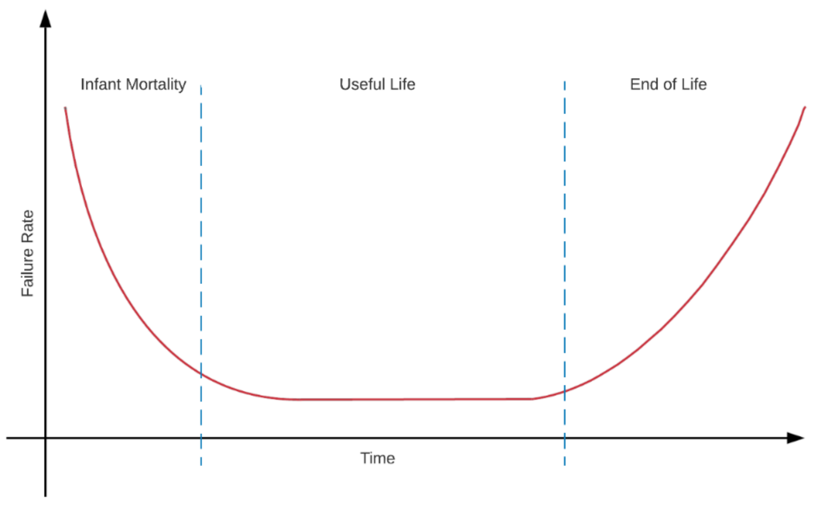


Figure 2.5, The bathtub curve for failure rate over time.

Figure 2.5 demonstrates the bathtub curve along with its three distinctive failure modes. The three main modes of failures can be described as follows:

1- Instant mortality: Failures within this phase are categorized with randomly distributed failures due to weakness or defects in the materials, components, or production process. The failure rate generally decreases into a stable/constant rate before it reaches the second phase rapidly.

2- Useful life (or constant failure rate): The failure rate in this phase is approximately constant and occurs randomly within a given time interval and thus can be described with a poison distribution.

3- End of life (or wear-out failures): In the final phase, the failure rate increases due to item aging, wear-out, or fatigue.

2.2.5 Common Failure Metrics

The *Mean Time to Failure* (MTTF) metric can be defined as the time point to the system's first failure.

$$MTTF = \frac{1}{N} \cdot \sum_{i=1}^{N} t_i \qquad (2.5)$$

As illustrated in Equation 2.5, the estimated time can be calculated by measuring the time to the first failure t_i for several N identical components.

$$MTTF = \int_0^\infty e^{-\lambda t}\, dt = \frac{1}{\lambda} \tag{2.6}$$

The MTTF can also be expressed in reliability measures, as illustrated in Equation 2.6. The MTTF can be described as the inverse of the failure rate when the exponential failure law is assumed.

The *Mean Time to Repair* (MTTR) can be defined as the expected time until a specific component is repaired.

$$MTTR = \frac{1}{N} \cdot \sum_{i=1}^{N} t_i \tag{2.7}$$

The required time t_i to repair the i^{th} component among N identical components can be calculated with Equation 2.7. The relation between the MTTR and the operational cost is inversely proportional, i.e., frequent hardware repair increases the operational cost but decreases the MTTR.

The *Mean Time Between Failure* (MTBF) is defined as the sum of the elapsed time for MTTF and MTTR (i.e., $MTBF = MTTF + MTTR$).

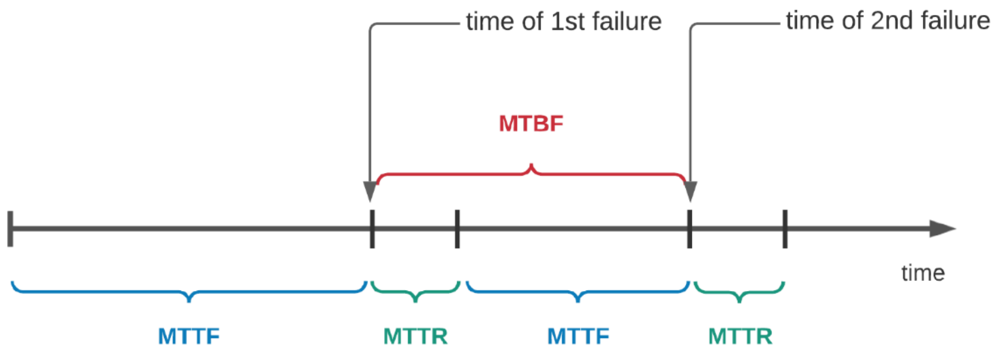


Figure 2.6, The relation between the standard failure metrics.

As illustrated in Figure 2.6, the MTTF is often approximated to MTBF as the MTTR is considered to be relatively small. [19]

2.3 Cloud Computing

According to [20], Cloud computing can be referred to as a combination of software and datacentre hardware that collaborates to deliver applications over the internet as a service.

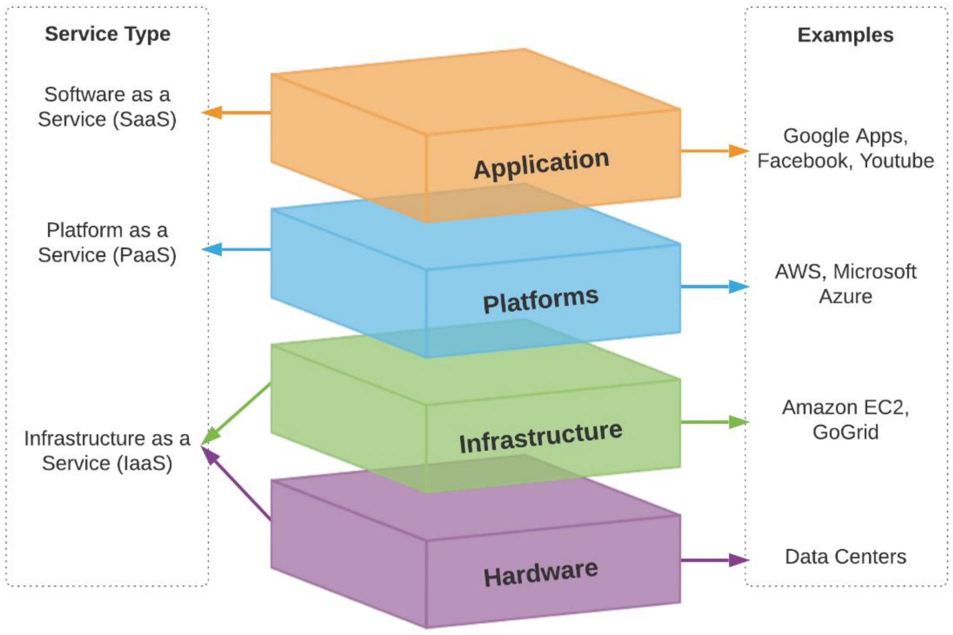

Service Type
Software as a Service (SaaS)
Platform as a Service (PaaS)
Infrastructure as a Service (IaaS)
Application
Platforms
Infrastructure
Hardware
Examples
Google Apps, Facebook, Youtube
AWS, Microsoft Azure
Amazon EC2, GoGrid
Data Centers

Figure 2.7, The main cloud computing services with examples.

As illustrated in Figure 2.7, the services provided on the clouds can be divided into four different layers: a hardware (or data center) layer, infrastructure (or virtualization) layer, platform layer, and application layer.

The hardware layer is responsible for providing the required components (e.g., servers, routers, switches, power, and cooling systems) to be utilized by the subsequent layers. The infrastructure layer facilitates the available resources (e.g., storage or computing power) dynamically with the help of different vitalization mechanisms. The platform layer consists of a simplified interface using operating systems and application frameworks that allow interactions with the lower layers. In the application layer, the actual cloud applications are deployed. Traditionally, cloud computing resources are utilized in a service model where each layer can be accessed and used on-demand. These services can be described as Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [21].

2.4 Machine Learning

Machine learning (ML) is a discipline that studies different computer algorithms that can automatically learn new patterns using data-driven methods. The main objective of ML is to predict new items with high accuracy and efficiency. ML typically inherits many theories and methods established in computer science (i.e., statistics, probability, and optimization). This also means that the gathered data is subjected to pre-processing and data analysis to enable the algorithms to perform more accurately. [22]

2.4.1 Key Elements of ML

According to [22], there are some terminologies need to be addressed when dealing with ML tasks. The main terminologies are *Examples* (or dataset) which refers to the items or data points used for learning or evaluation, *Features* that can be defined as the example's attributes, *Labels* which are referred to the target attribute that can be represented as values or categories, *Hyperparameters* that can be defined as the parameters that are not essentially defined as an input by the learning algorithm, *Training samples* which refers to the examples used for the learning algorithm, *Validation samples* which can be defined as the examples used in optimizing learning algorithms that uses labelled examples. *Testing samples* which can be defined as the examples used to evaluate the performance of the learning algorithm. It is also worth noting that the testing sample must be treated as a production environment (i.e., not used in the learning and validation stages). Finally, loss function refers to the process that measures the differences between the predicted value and the actual value.

2.4.2 Learning Paradigms

As suggested in [22], there are certain types of tasks that ML is considered to be a perfect candidate to accomplish. Those tasks require specific types of learning mechanisms based on the provided inputs and desired results.

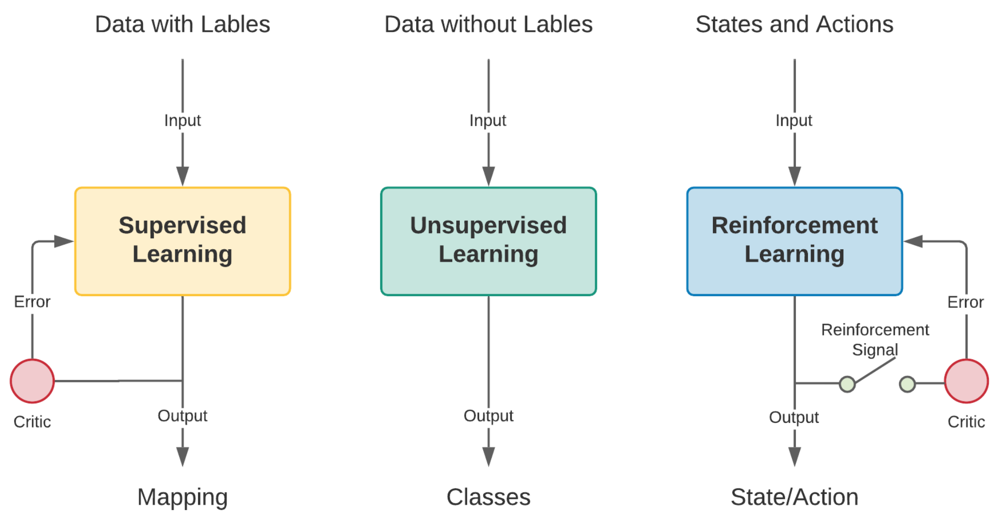


Figure 2.8, Common learning paradigms in ML.

Figure 2.8 demonstrates those learning paradigms with their main differences and components.

According to [22], the following describes some of the most common ML paradigms along with their typical use scenarios:

Supervised Learning: In supervised learning, the ML algorithm expects a set of labeled examples in the training sample. Here, the prediction will be made on the unseen data points. Some of the most common tasks that use supervised learning are classification, regression, and ranking problems. In classification tasks, each item is assigned with a specific category that can then be used to make further predictions. The number of categories might vary from two to a few hundred, based on the type of task that is assigned (e.g., anomaly detection, image classification, etc.). In contrast, regression tasks focus on making a future prediction of real values rather than choosing a suitable category as in classification tasks. One typical example of such tasks are *time series forecasting* tasks, where future predictions are based on historical data. The evaluation of this kind of task depends on measuring the magnitude between the predicted value and the actual value for a specific item. In ranking tasks, the goal is to order the items based on specific predefined rules. These kinds of tasks are currently used in many search engines where the search is optimized by showing the most relevant queries.

Unsupervised Learning: In contrast to supervised learning, the training samples in unsupervised learning don't contain any labels and the predictions are still been made on the unseen data points. Some common tasks that make use of such a learning method are clustering and dimensionality reduction. In clustering tasks, the main goal is to partition many items into homogenous groups or subgroups (e.g., customer segmentation). On the other hand, in dimensionality reduction tasks, the main focus is to reduce the complexity of an item by representing it with lower dimensions while preserving the essential information (e.g., Big data visualization).

Reinforcement Learning: In reinforcement learning, the training and testing phases continuously operates to obtain the desired results. The ML algorithm tends to interact with the observed environment within a pre-set number of rules. Here, predictions are made in the form of actions that can be evaluated in the form of reward or punishment. The satisfaction of such a model relies on maximizing the rewards or minimizing the punishments. Reinforcement learning is often used in real-time applications such as robot navigation, real-time decisions, and skill acquisition.

While the listed algorithms are considered to be common for many ML scenarios, some complex tasks might require other intermediate solutions that combine different methods and algorithms.

2.4.3 Neural Network

Neural Network (or Artificial Neural Network) is a specific set of ML algorithms that are heavily inspired by the learning mechanisms found in biological neural networks.

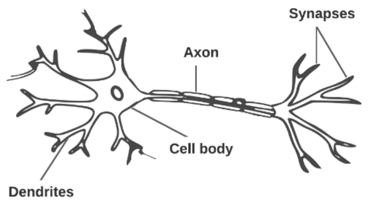


Figure 2.9, Biological Neural Network

As demonstrated in Figure 2.9, the human nervous system, for instance, contains Neurons that essentially consist of axons, dendrites, and synapses. The main function of axons and dendrites is to connect the neurons to their adjacent ones forming a network of neurons. The gaps formed between those connections are referred to as synapsis. In the process of learning external or internal information, each synapsis adapts by adjusting the strength of each connection. [23]

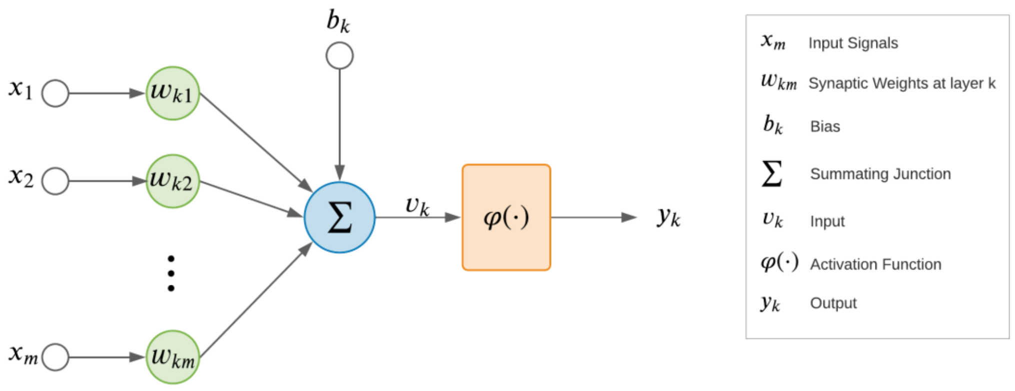


Figure 2.10, Structure of a typical Artificial Neural Network.

As explained in [24], neural networks consist of a number of artificial neurons capable of finding different patterns within the input data to determine the final output. As shown in Figure 2.10, the basic scheme of artificial neurons consists of a number of input signals x_j, synaptic weights w_{kj}, bias b_k, Activation function $\varphi(\cdot)$, and output y_k. It is important to define the subscripts of the weights as k refers to the neuron in question and j refers to the input signal that will be subjected to weight multiplication.

In the process of learning, the strength of each input signal is manipulated by the magnitude of each respective weight i.e., a positive value of weight refers to an excitatory connection and a negative value refers to an inhibitory connection. All weighted input signals are summed up in the summing junction and processed by the activation function. In this stage, the summed is adjusted to fit the desired output by transforming it to a manageable closed unit interval (e.g., $[-1, 1]$ or $[0, 1]$). In most cases, a bias is used to set a threshold for the weighted sum. This threshold can then be used to determine the value that triggers the activation function to generate meaningful outputs.

$$u_k = \sum_{j=1}^{m} w_{kj} + x_j \tag{2.8}$$

$$y_k = \varphi(u_k + b_k) \tag{2.9}$$

In mathematical terms, the components of the artificial neuron can be described as in Equation 2.8 and Equation 2.9.

2.4.4 Convolutional Neural Network

Convolutional neural networks (CNN) are neural network architectures optimized for automatically learning different features and patterns of spatial hierarchies. [25]

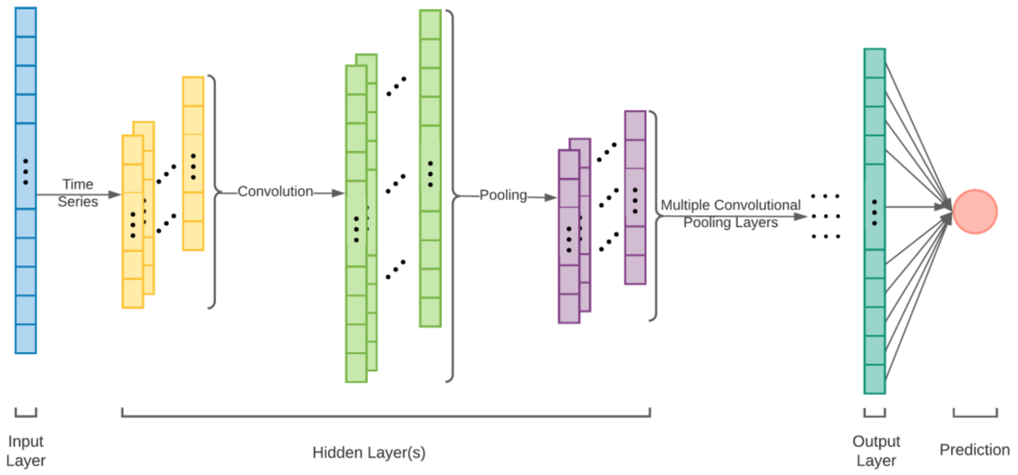


Figure 2.11, Structure of convolutional neural network.

As demonstrated in Figure 2.11, a basic CNN consists of an input layer, output layer, and a pattern recognition layer known as a convolutional layer. The input layer is mainly where the data processing begins. Here, images or sequential data are fed into the network to be processed and analyzed in the convolutional layer. Once the data arrives at the convolutional layer, different patterns are extracted from the input data using a data transformation technique.

$$X_j^l = f\left[\sum_{i \in M_j} (X_i^{l-1} * K_{ij}^l + b_j^l)\right] \tag{2.10}$$

The transformation consists of different filters that map each feature by convolving across each input data using a specialized type of linear operation demonstrated in Equation 2.10. Where X_j^l represents the j^{th}feature map at layer l, M_j represents the input data (or image) set, X_i^{l-1} represents the i^{th}feature map at the previous layer $l-1$, the convolution operation $(*)$, K_{ij}^{l-1} represents the connecting filter between X_i^{l-1} and X_j^l, b_j^l represents the applied bias at the layer l, and the $f[\cdot]$ represents the utilized activation function (e.g., sigmoid, tanh, ReLU, etc.).

As a result, different patterns and features are extracted from the input data. In CNN, features can be extracted in the form of one dimension (e.g., sequences of observations in time-series) or more (e.g., shapes, textures, and edges in images). The final calculation will be represented probabilistically in the output layer. [26]

2.4.5 Recurrent Neural Network

Recurrent Neural Networks (RNN) is a common feed-forward neural network that typically operates with sequential data such as time-series data or text sentences. Here, the input data comes in the form of (x_1, x_2, $\ldots, x_n$) where x_t denotes to the datapoint with d-dimensions at time-stamp t.

The prediction mechanism in sequential data is dynamic, where its prediction depends on all past information datapoints. This requires the architecture to allow every new datapoint to interact with the hidden internal states that were formed by the previous inputs. There are three main elements that enable the RNN to make predictions with time-stamps.

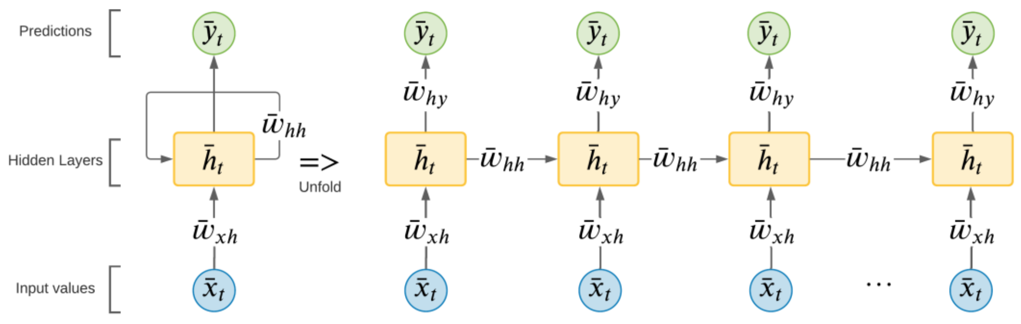


Figure 2.12, A recurrent neural network with unfolded form.

These elements are input data $\bar{x}_t$, hidden internal state $\bar{h}_t$, and output data $\bar{y}_t$ and illustrated in Figure 2.12. The input data are used to feed the RNN with new information where the hidden state $\bar{h}_t$ is activated every time it reads an input hidden states are able to adapt to new information by making use of the previous inputs. The output $\bar{y}_t$, on the other hand, represents the predicted forecast of $\bar{x}_{t+1}$. In some tasks are required to output any results (e.g., time series values) but rather is satisfied by an evaluation of the final hidden state to make predictions. As an example of such tasks are classification tasks, where the final state is utilized to make a prediction on a category of interest.

$$\bar{h}_t = f(\bar{h}_{t-1}, \bar{x}_t) \tag{2.11}$$

The relation between those elements is represented in Equation 2.11. Here, the new hidden state $\bar{h}_t$ is based on the function of the previous hidden states $\bar{h}_{t-1}$ and the current input $\bar{x}_t$. [23] In RNN, one learning iteration consists of a forward and backward pass. The process of calculating the output from the available inputs is known as forward pass, and the process of adapting the neurons to fit a certain problem is known as a backward pass (i.e., backpropagation).

It is important to note that all neurons in RNN are subjected to backward pass as they all contribute to the final output. This leads to major challenges in updating the weights in the early layers of the neural network. This problem is referred to *vanishing gradient problem* where the weights are not essentially updating due to small gradients that don't contribute to any learning.

The basic RNN also suffers from short memory that doesn't allow it to keep track of the early sequencies of inputs and thus forgets long-term dependencies. The short memory problem was addressed and mitigated by proposing some modified RNN architectures such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). The main difference between these neural networks and basic RNN is that they use some internal mechanisms that allow them to extract the most relevant information.

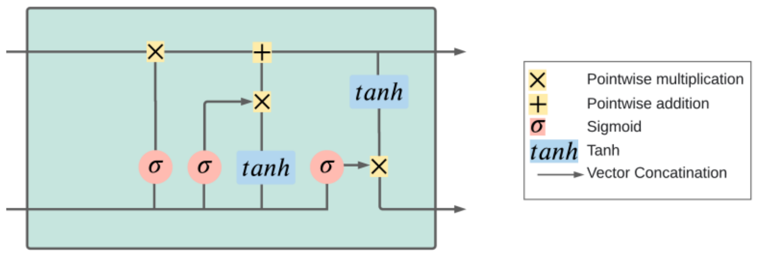


Figure 2.13, A single LSTM cell with operations.

In the case of LSTM, the neural network consists of memory cells that in turn contain three gates, namely, the *forget gate*, the *input gate* and the *output gate,* and can be illustrated in Figure 2.13. The forget gate's main function is to save memory by filtering the irrelevant information from further feeding to the subsequent hidden states. All relevant information passes through the input gate in the form of hidden states. Those hidden states will be subjected to a sigmoid and tanh function.

The sigmoid function will determine the level of importance of the received state, and the tanh function will transform the values into the range $[-1, 1]$. Both acquired results are then multiplied to determine which information to keep from the tanh function. At this stage, the new cell state can be calculated by multiplying the current cell state with the forget vector and then adding the product with the acquired value from the tanh function. In the output (and final) gate, predictions of the new hidden state are being made. This gate acquires its value by multiplying the current transformed input with the previous hidden state that was acquired from the aforementioned step. [27]

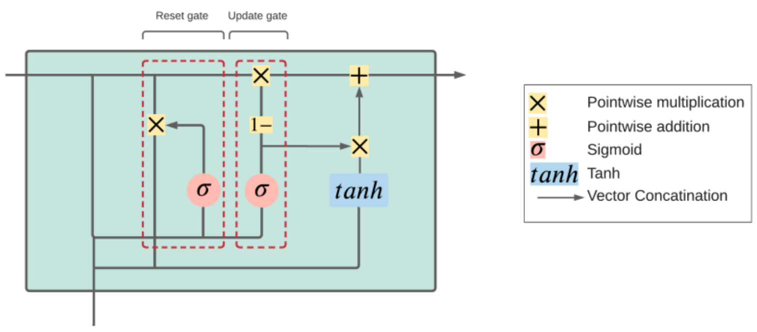


Figure 2.14, A single GRU cell with operations.

Unlike LSTM, GRU only uses two gates namely the reset gate and the update gate as illustrated in Figure 2.14. In the reset gate, all irrelevant information is discarded, while in the update gate, the information importance is determined to use it as an input or to discard it. Another difference from LSTMs is that GRUs uses the available hidden states to update the cells' state. [28]

2.5 Related Work

This section describes four related pieces of research in the area of hardware failure detection and predictive maintenance.

2.5.1 Concept Drift Detection and Adaption in Big Imbalance Industrial IoT Data Using an Ensemble Learning Method of Offline Classifiers

C. Lin et al. proposed in [29] a collaborative method by the name of *dynamic AdaBoost.NC with multiple sub-classifiers for imbalance and drifts* (DAMSID) for enhancing the maintenance.

The method focuses on providing a condition-based maintenance approach for Industrial IoT components by detecting concept drift through data streams. The method makes use of ensemble learning techniques that allows different models to operate collaboratively on offline classifiers. The proposed solution was developed in three different stages. One stage aimed to build the ensemble classifier; one stage aimed to deploy the improved classifier using AdaBoost.NC and SMOTE method; and one improves the detection performance by using Linear Four Rates method. The proposed classifier was able to detect all proposed concept drifts with 94% accuracy.

This related work shares a number of similarities with the presented thesis in terms of providing a drift fault detection mechanism using sensor data stream and supervised learning. The differences, however, lie in providing a scalable predictive maintenance approach for drift faults detection as well as providing an approach for labeling the dataset for learning purposes.

2.5.2 Data-driven predictive maintenance planning framework for MEP components based on BIM and IoT using machine learning algorithms

J.C.P. Cheng et al. suggest in [30] a maintenance strategy that improves the efficiency of facility maintenance management (FMM) by proposing a data-driven predictive maintenance framework.

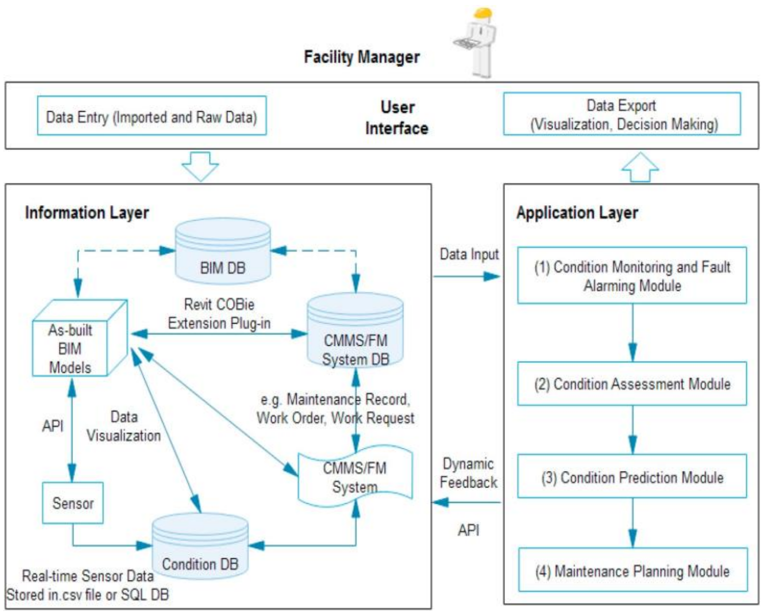

Facility Manager
Data Entry (Imported and Raw Data)
User Interface
Data Export (Visualization, Decision Making)
Information Layer
BIM DB
As-built BIM Models
Revit COBie Extension Plug-in
CMMS/FM System DB
e.g. Maintenance Record, Work Order, Work Request
API
Data Visualization
CMMS/FM System
Sensor
Condition DB
Real-time Sensor Data Stored in.csv file or SQL DB
Data Input
Dynamic Feedback
API
Application Layer
(1) Condition Monitoring and Fault Alarming Module
(2) Condition Assessment Module
(3) Condition Prediction Module
(4) Maintenance Planning Module

Figure 2.15, An overview on the proposed solution. [29]

As illustrated in Figure 2.15, an information layer and application layer were developed to enhance the FMM with data provided by the IoT devices and Building Information Modeling (BIM). In the information layer, the data was gathered and integrated with the application layer which in turn handles four different modules. Namely, condition monitoring and alarms, condition assessment, condition prediction, and maintenance planning.

The proposed approach makes use of two different ML prediction algorithms namely ANN and SVM. The results emphasize the importance of continuously updating the data to gain effective predictions of the conditions. The suggested approach and this thesis discover different approaches for applying data-driven predictive maintenance. Both studies make use of different ML algorithms to increase maintenance efficiency.

There are however a number of differences that signify each study where the study provided by J.C.P. Cheng et al. focuses on providing a condition monitoring scheme for the facility as a whole using generated data from IoT and BIM. On the other hand, this thesis focuses on providing a monitoring scheme for the installed IoT devices through data streams. Another significant difference is the choice of the ML algorithms that use different regression prediction models (i.e., ANN and SVM) where this thesis uses time series forecasting and classification algorithms with LSTM, CNN-LSTM, and GRU.

2.5.3 Faulty Sensor Detection, Identification and Reconstruction of Indoor Air Quality Measurements in a Subway Station

M. Huang et al. propose in [31] a sensor fault detection mechanism that increases the performance of the air quality monitoring tool. The approach uses an adaptive network-based fuzzy inference system (ANFIS), to detect non-linear sensor faults, and a structured residual approach with maximum sensitivity (SRAMS), to identify and reconstruct sensor faults sequentially.

The detection mechanism uses four different metrics to predict the sensor fault. Those identification metrics are filtered squared residual, generalized likelihood ratio, the cumulative sum of residuals, and cumulative variances index. The approach was tested in a real subway environment where the solution was able to detect effectively four different types of sensor faults namely sensor drift, bias, precision degradation, and complete failures.

The presented solution and this thesis focus on providing a data-driven approach for real-time sensor fault detection. The main difference is that Huang et al. suggest an approach that uses statistical methods with no interest in scalability. This thesis, on the other hand, proposes a deep learning approach that uses a scalable platform.

2.5.4 Automatic Sensor Drift Detection and Correction Using Spatial Kriging and Kalman Filtering

D. Kumar et al. addresses in [32] the problem of sensor drift and bias that often occurs in Wireless Sensor Networks (WSN). The research has also proposed a framework for automatic sensor drift detection and correction using Kriging-based interpolation and Kalman filters.

The Kriging-based interpolation was used to detect sensor drifts through sensor readings of neighboring sensors and the Kalman filters were then used to estimate the level of drifts. Here, the estimation was based on applying a smooth sensor drifts to the original reading using a mathematical model that utilize Gaussian noise.

As a result, the proposed solution was able to detect smooth sensor drifts and bias from sensor readings. In addition, the developed system was able to scale upon bigger WSN without any significant tradeoffs when comparing with traditional averaging based interpolation methods.

The research paper and this thesis have aims to provide a sensor fault detection mechanism by utilizing the sensor readings. However, the differences lie in the methodology as D. Kumar et al. detect sensor drifts and bias by monitoring neighboring sensors using Kriging-based interpolation and Kalman filters. On the other side, this thesis investigates the use of RNN in detecting sensor drifts from the sensor data stream itself.

3 Methodology

In this chapter, the chosen scientific methods and thesis milestone are presented and motivated for the fulfillment of the overall aim in this thesis.

3.1 Scientific Method Description

The scientific basis in this thesis will be carried out with a typical research workflow that consists of six different stages.

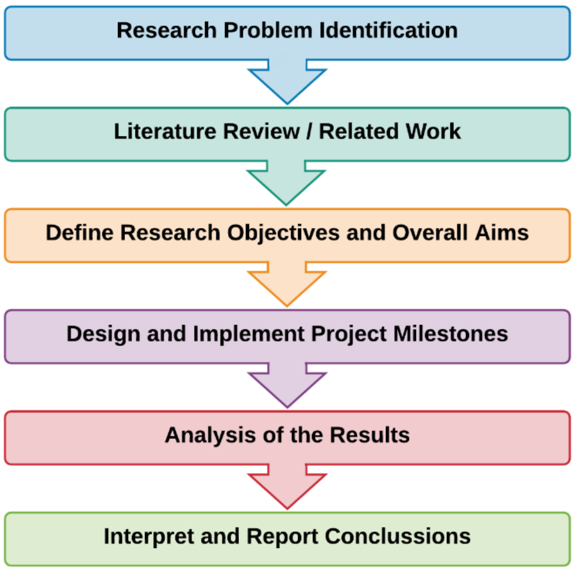

Research Problem Identification
Literature Review / Related Work
Define Research Objectives and Overall Aims
Design and Implement Project Milestones
Analysis of the Results
Interpret and Report Conclussions

Figure 3.1, Research workflow.

As illustrated in Figure 3.1, these stages are research problem identification, literature review and related work, defining research objectives and overall aims, designing and implementing thesis milestones, analyzing the results, and finally interpreting and reporting conclusions.

In the research problem identification stage, the main goal will be to identify the characteristics of the specific use-cases and research problems in the fields of smart agriculture and predictive maintenance. Once the problem is identified and motivated, a literature review will be conducted to acquire the required knowledge on relevant theories and state-of-the-art solutions that could potentially be a part of the proposed solution.

The next stage will be defining the main overall aims of this thesis in a list format that will be used in the subsequent stage. In the design and implementation stage, the overall aims will be used in forming the characteristics of the quantitative research in this thesis. This will be achieved by shaping the implementation methods and the key performance metrics that will further be analyzed.

In the analysis of the results stage, the observed quantitative data will be analyzed with statistical methods with the help of standard deviation (SD), confidence interval (CI), one-way Analysis of Variance (ANOVA), and Tukey's Honest Significant Difference Test (Tukey's HSD) test to estimate any significances in the results. Finally, all outcomes from this thesis along with the ethical and societal aspects, will be discussed and concluded, and documented in this report.

3.2 Thesis Method Description

The satisfaction of each implementation goal in this thesis will be carried out in five different milestones.

3.2.1 Milestone 1: Literature Review

The first milestone can be satisfied by finding the three most prominent state-of-the-art solutions that could be utilized in predictive maintenance with time series. The literature review will also cover the related fields to hardware maintenance of IoT devices and fault detection mechanisms with ML. This will be achieved primarily by performing a literature review on the most cited and peer-reviewed systematic review articles as well as some state-of-the-art surveys on Google Scholar. The reason why Google Scholar will be used for publication search is due to its ability to make both extensive and specified searches of scientific publications from different scientific databases.

The collected articles will be categorized and analyzed to gain a solid and broad understanding of the latest work that has been conducted in the research area. The literature review will occasionally be complemented by a comprehensive review of linked scientific publications on cloud computing and predictive maintenance. This will be used to extend the knowledge on the feasibility of utilizing cloud computing platforms in enabling IoT applications. This part of the thesis will also result in a broad understanding in the areas of IoT, reliability theory, ML, neural networks, and cloud computing, and reviewed briefly in Chapter 2.

3.2.2 Milestone 2: Design

The second milestone will be fulfilled by designing a solution that fits the needs of smart agriculture environments. This will be carried out by identifying the characteristics of hydroponic agriculture and failure detection mechanisms. This step is essential in order to make sure that the formulated problem statement and overall aim will be satisfied in this work.

A data pipeline will be designed to manage the data streams in two different stages, namely, data generation and data forecasting and classification. The data generation stage will be carried out by a low-cost temperature and humidity sensor that will take measurements at fixed time intervals.

The data forecasting and classification will be designed to be deployed at the cloud computing entity and will be created according to the CRISP-DM model. This stage will be handling the received data streams by preprocessing the data in order to maintain a more accurate result. Once the data streams are prepared, the ML model will be executed to make the required multi-step forecasts and classifications. In addition, a proof-of-concept will be deployed in a cloud computing environment to analyze its feasibility in terms of response time.

3.2.3 Milestone 3: Implementation

The satisfaction of the third milestone will be realized by implementing the predefined data pipeline (i.e., the three data management stages).

The data generation will be solely carried out by the temperature and humidity sensors in the first stage. Here, the sensor will be deployed in a generic indoor environment due to limitations in acquiring a natural hydroponic environment. The sensors will also be connected to a typical IoT device that operates as a gateway to carry out the data flow to the cloud platform. In the second stage, the data will be stored in a cloud-based database and then pre-processed to carry out the subsequent tasks. Once the data is prepared, the prediction models will be developed in two different stages. The first stage will be the multi-step forecasting task and the second stage will be the classification stage.

It is worth noting that the size of interval in multi-step forecast will be variating to be able to choose the most optimized size. Once the models are be developed a fine-tuning step will take place by adjusting their hyperparameters (i.e., no. of Epochs, batch size, no. of layers, hidden units, activation functions, hidden units, and drop rate).

Finally, the proposed solution will be deployed in a cloud computing environment to detect failures by making predictions, the system will also be analyzed in terms of the time required to handle the prediction requests.

3.2.4 Milestone 4: Measure

The fourth milestone will be satisfied by taking measurements concerning both the performance of the system and the prediction models. The performance of the prediction models will be measured with the help of different accuracy metrics. In the case of the forecasting model, the performance of each interval size will be measured by calculating their respective total Root-Mean-Squared Error (RMSE).

In contrast, for the classification model, the performance will be obtained by calculating the predictions F1-score and accuracy. In the context of the system's performance, the operability of the final solution will be measured by taking the average response time. Here, the system's ability to execute the aforementioned three stages and handling loads of requests will be compared against an empty request that represents the ground truth of the response time.

3.2.5 Milestone 5: Evaluate

In order to satisfy the fifth milestone in this thesis, the developed system will be evaluated statistically to determine its overall performance. Here, all results obtained from the proposed solution (i.e., average response time and accuracy metrics) will be examined several times. As mentioned in Section 3.1, all variations in the results will be analyzed by calculating the SD and CI of 95%. Consequently, all measurements from the forecasting and classification steps will be subjected to an ANOVA test with a p-value of 0.05 to detect any significant differences. This procedure will be carried out by initially assuming a *null hypothesis* that of no significant difference at the level of 0.05 and then proof that overwise by observing the overall significance in the measurements.

Once the overall significance is detected, the measurements will also be analyzed pairwise using Tukey’s HSD test to determine the exact occurrence of significances. This will provide a broader understanding of the reliability of the measurements and whether the proposed solution can be deployed in a production environment for this particular scenario.

3.3 Thesis Evaluation Method

The thesis will be evaluated based on the fulfillment of the overall aim. In detail, the thesis scientific goals and thesis milestones will be reviewed in terms of their satisfaction, contribution to the field of predictive maintenance and ML, encountered limitations, ethical and societal considerations, and future directions.

The evaluation of the thesis milestones will cover all essential elements that shape this thesis. The evaluation of this thesis will also cover topics like decisions regarding the design, chosen solution, implementational methods, and choice of the evaluation metrics.

4 Choice of Cloud Platform

This chapter presents a comparison of the most prominent cloud service providers. The comparison focuses on the recent services and functionalities that enable the deployment of the solution. More specifically, the cloud platform should be able to support services such as computing, storage, security, IoT, and ML. Therefore, this comparison will be used as a basis for making decisions for technical choices.

4.1 Amazon Web Services

Amazon Web Services (AWS) is a public cloud service platform developed by Amazon. The platform provides different services that could be utilized in industrial and non-industrial applications. [33] The following sections provide a description of the most relevant services with examples that could potentially be used in this thesis.

4.1.1 Computing Services

AWS offers a broad range of computing functionalities that can be flexibly customed based on the task to be performed. There are thirteen different services that facilitate the computing offerings for different use cases. The following is a comprehensive description of some of the most relevant services to this thesis:

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides flexible access to computing resources. The service can be easily and swiftly initiated to perform the assigned computing tasks. Amazon EC2 offers also fault tolerance functionalities that isolate any occurring failures. The cost of Amazon EC2 might vary based on three different price models, namely on-demand (i.e., pay only for the utilized compute capacity), reserved instance (i.e., pay for the reserved capacity at a specific region), and spot instance (i.e., unlock spare capacity available).

AWS Lambda assists the developers by reducing the complexity of setting up servers and other resources to run a specific code. AWS Lambda functions can be virtually performed on all types of deployed applications. In AWS Lambda, payment is only required when a function is running either through event triggers or direct function calls. [34]

4.1.2 Storage Services

There are seven different options that provide storage of data in a variety of formats at the cloud platform. The following describes the most relevant service that can potentially be used when implementation:

Amazon Simple Storage Service (Amazon S3) is a data storage service that can scale without any compromises in terms of functionality, security or complexity. Some typical use-cases of S3 are the storage of data generated by IoT devices, data analytics, and data backup. [35]

4.1.3 Security Services

Currently, there are seventeen security services that are offered by the AWS cloud platform. These services are meant to manage the security and compliance of the services as well as the identity of the end-users. The following is a description of the most relevant provided services:

AWS Identity and Access Management (IAM) manages the level of user permissions and access to the provided AWS resources.

AWS Certificate Manager is used to create certificates that can maintain a secure interaction between the AWS services and the connected devices and applications. This can be performed by functionalities that allow the service to facilitate, manage, and install Secure Sockets Layer/Transport Layer Security (SSL/TLS) certificates. [36]

4.1.4 IoT Services

There are mainly twelve different IoT-related services that are offered by the AWS cloud platform. The following describes the most relevant services that could be useful when implementing the proposed solution:

AWS IoT Core is the main interface for connecting the IoT devices to the cloud platform. The service allows the platform to securely gather, process, and analyze the data formed by the connected IoT devices. Moreover, AWS IoT Core can create complete IoT applications by allowing the IoT devices to integrate with different AWS services such as AWS Lambda, Amazon SageMaker, and Amazon S3.

AWS IoT Device Management provides a service that allows the deployment of a massive number of IoT devices. This could be achieved by functionalities that allow bulk or individual connections, device monitoring, over-the-air updates, and managing device permissions. [37]

4.1.5 ML Services

AWS also provides an array of ML services. There are currently a total of eighteen different services that are used according to the task in hand. The following presents the most relevant services that could be considered as a part of the developed solution:

SageMaker is a service that provides a fully managed platform that allows the development of ML algorithms. SageMaker contains functionalities that allow the developers to create, train, and integrate ML models into production environments. [38]

4.2 Microsoft Azure

Microsoft Azure (or Azure) is also a cloud platform that provides many services such as computing and storage services. Similar to AWS, Azure is also considered to be one of the leading cloud service providers. [39] The following sections describe some potential services that could be utilized in this thesis.

4.2.1 Computing Services

Azure offers a wide variety of computing services. The services can easily scale according to the required capacity of the application. [40] The following list provides the most relevant services that can be used as part of the proposed solution:

Azure Virtual Machines (VM) provides computing resources in both Linux and Windows virtual environments. VM services were built to scale and managed according to application needs. The virtualization property allows the developers to only focus on building applications by eliminating any hardware purchases or maintenance. [41]

Azure Virtual Machine Scale Sets is a service that increases the availability of VMs by allowing them to scale up/down based on the demand. This can be done by utilizing a number of load balancers on the top of each virtual machine. This property allows the development of large-scaled projects such as big data and container loads. [42]

Azure Functions provide serverless computational power on the cloud platform that allows the developers to access updated infrastructures and resources on-demand. The service often runs reactively where a certain event needs to take place in order to trigger the function. [43]

4.2.2 Storage Services

Azure provides a variety of storage services that can easily scale and maintain. This service is compatible with storing different data structures such as objects, files, queues, and tables. The following service is considered to be prominent to use in the proposed solution:

Blob Storage enables the storage of unstructured objects such as text and binary files at the cloud. The Blobs are optimized to store different types of data such as images, documents, audio, and video files. [44]

4.2.3 Security Services

Azure offers many security functionalities and security technologies that can be categorized based on their use; these categories are General Azure security, Storage security, database security, identity, and access management, Backup and disaster recovery, and Networking. [45]

4.2.4 IoT Services

Currently, Azure provides a number of services and functionalities that allow the deployment of comprehensive IoT solutions as well as asset management. The following describes the most relevant service that can be used in the proposed solution:

IoT Hub is a service that enables bidirectional communication between the IoT application and their connected devices. This service contains functionalities such as device-to-cloud telemetry, file upload from devices, and request-reply methods that can be used to control remote control of the connected devices. [46]

4.2.5 ML Services

Azure Machine Learning provides a cloud-based environment for ML. The environment allows training and deploying supervised and unsupervised algorithms as well as different model types such as deep learning. This service can easily be integrated with other computing, networking, and storage services. [47]

4.3 Comparison of Cloud Platforms

This comparison in this section will be based on four criteria, namely the availability of the available cloud services, communication protocols, average request rate in handling 1,000,000 requests, and programming language support.

Table 4.1, Cloud platform comparison.

	AWS	**Azure**
Computing	Yes	Yes
Storage	Yes	Yes
Security	Yes	Yes
IoT	Yes	Yes
ML	Yes	Yes
Communication Protocols	HTTP, MQTT, MQTT over WebSockets, and LoRaWAN	MQTT, MQTT over WebSockets, AMQP, AMQP over WebSockets, HTTPS
Avg. Request/s	1025.29 [48]	1119.41 [48]
Programming Language Support	Java, JavaScript, C#, PHP, Python, Ruby, Go, and C++	C#, JavaScript, F#, Java, PowerShell, Python, and TypeScript

As illustrated in Table 4.1, the general specifications partially similar in both cloud platforms. The main differences occur in the available communication protocols, average response time, and the supported programming languages.

4.4 Chosen Platform

Both AWS and Azure offer a wide variety of services that shows promising potential in facilitating the implementation of the final solution. Both platforms make use of the most common application layer communication protocols, which enable the implementation of many different IoT scenarios. On the other hand, AWS offers a wide variety of functionalities that ease the process of experimentation and evaluation. The service also contains a wide variety of educational resources such as documentation, tutorials, and community for the developers. The platform of choice for this thesis project will be AWS, as they tend to contain all the required services and functionalities.

5 Implementation

This chapter describes in detail the process of constructing the proposed solution and its related evaluation procedure.

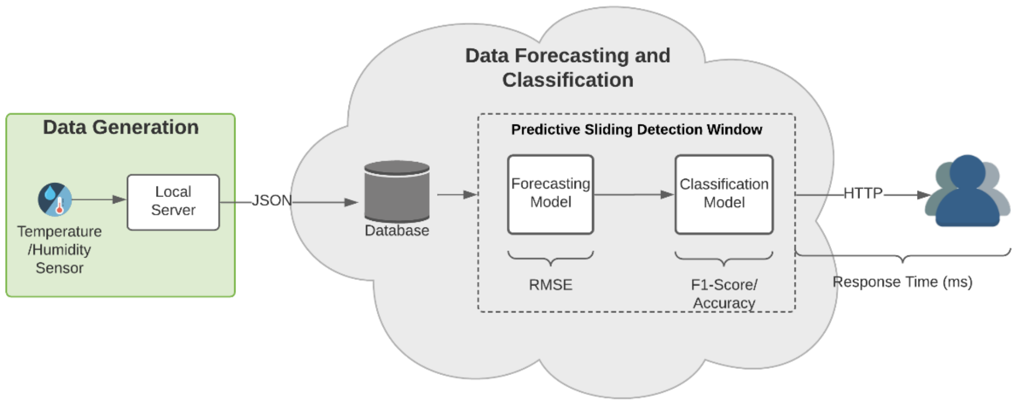


Figure 5.1, An overview of the proposed solution.

As illustrated in Figure 5.1, the implementation was arranged in a two-layered setup (i.e., Data Generation and Data Forecasting and Classification) where each layer maintains a certain functionality in order to satisfy the thesis's overall aim.

5.1 Data Generation

The data generation was mainly carried out with a single temperature and humidity sensor connected to AWS cloud services. The local server was configured on a Raspberry pi model 3B, hereinafter will be referred to as RPi, where it acted as a middleware between the sensors and the AWS cloud. The sensor was directly deployed to the local server where a script was configured to read the sensing values once every 60 seconds. On the other hand, the local server was connected to the AWS cloud wirelessly. This was achieved by configuring the communication between the local server and the AWS IoT Core.

The local server was initially registered as an *IoT Thing* in the AWS IoT Core, then authenticated using AWS's certificate authority, and finally attached to a policy that allows the IoT Thing to publish the sensing data to the cloud. Once the configuration was completed, the local server was able to authenticate itself using its provided private key and then publish its sensing data in JavaScript Object Notation (JSON) format to AWS IoT Core.

All received sensor readings were then aggregated using Amazon's Kinesis Firehose service in a buffer interval of 900 seconds. This was made in order to reliably load streaming data into data lakes configured on an S3 Bucket storage unit.

5.2 Data Forecasting and Classification

This layer is considered to be the core of this thesis, where the gathered data was forecasted and classified. This was achieved by utilizing a conventional workflow known as Cross Industry Standard Process for Data Mining (CRISP-DM).

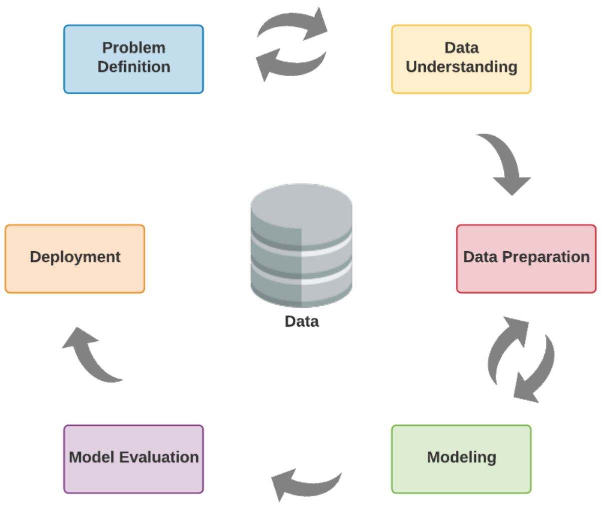


Figure 5.2, Data forecasting and classification workflow.

The workflow is illustrated in Figure 5.2 and included activities such as problem definition, data understanding, data preparation, modeling, evaluation, and deployment. The following sections describe the activities in more detail and provide an overview of the developed preventive maintenance algorithms.

5.2.1 Problem Definition

This activity was influenced by the original problem motivation in this thesis. The thesis aims to investigate an approach for increasing the efficiency of maintaining the hydroponic sensors with ML. In addition, the thesis aims to discuss the reliability and accuracy of the proposed solution in detecting sensor drift failures from sensor data streams. These goals could be fulfilled by enhancing the hydroponic sensors with an anomaly detection mechanism that relies on sensor readings.

To increase the efficiency of detecting anomalies, the proposed approach was assisted with forecasting and classification ML models. The developed forecasting model makes use of univariate features (e.g., temperature or humidity) to predict several future time steps and hence can be referred to as a *univariate multi-step time series forecasting model*. The classification model, on the other side, was used to detect failures in both actual and forecasted values.

5.2.2 Data Understanding

The gathered dataset was obtained from the first layer where it consists of four different attributes. The total number of data points in the dataset was 91774, where a single sensor reading was made every one minute. The attributes of each data point were sensor type, the temperature level (in °C), the humidity level in percent, and the date and time. The provided dataset and the literature review reveal that predicting hardware failures from only sensor data streams can be quite challenging due to a lack of failure examples and computing resources. Therefore, providing a solution that represents the ground truth by utilizing historical data can be the most feasible approach for this thesis.

The available dataset will be representing the ground truth for failure-free sensors as the theory presented in Section 2.2.4 was assumed. Therefore, the success of the proposed solution will rely on two main factors, namely the performance of the prediction model as well as the following three assumptions:

1- The deployed sensors are in new (or optimum) conditions.

2- The dataset is reliable and doesn't contain any anomalies.

3- Failures due to wear-out are the types of interest.

5.2.3 Data Preparation

The raw data was initially processed on a local device in order to avoid extra charges from AWS services. In this stage, the collected raw data was downloaded from S3 and investigated for further understanding and analysis. The investigation revealed a few numbers of missing data points in the dataset. Those data points were simply deleted from the dataset as they didn't affect the modeling step.

On the other hand, all data points were converted into a format that can be effectively handled when creating and evaluating the models. The data points were, for instance, restructured into an array of JSON objects, converted sensor readings (i.e., temperature and humidity) into float datatype, and timestamps into the date-time format. The next stage concerns assuring the quality of the dataset. This was maintained by assuring the existence of all data points and the order of all data points according to the specified time series. Once the data cleaning was completed, the data points were handled to fit the respective modeling methods, i.e., forecasting and classification models.

In the forecasting task, the data points in the dataset were represented in hourly time-steps as every sixty minutes was represented by the mean value. At this point, the total number of data points was 775. This step was also useful to eliminate any outliers that might be caused by sudden incidents (e.g., opening the window, passing by a sensor, etc.) that could potentially affect the model's performance. The next stage was aimed to normalize the data points to ensure an effective model learning when training. The dataset at this stage was ready for modeling once the data points were divided into training and testing sets. Here, the dataset was split into eighty percent training set and the rest as a testing set.

The classification task made use of a similar procedure in data preparation for the forecasting task. Here, the data was also represented by hours and then divided into training and testing sets. However, the dataset was duplicated into three datasets resulting in a total number of 2316 data points. The first duplicate was used to represent the ground truth (i.e., reliable data points), the second and third datasets were modified to represent sensor drift faults with ascending and descending data points, respectively. The sensor drifting fault values were estimated by applying a Gaussian noise to each data point as proposed in the related work in Section 2.5.4.

$$d_{i,k} = d_{i,k-1} \pm v_{i,k}, \qquad v_{i,k} \sim N\,(0, 1) \tag{5.1}$$

The drift was applied to the ascending and descending datasets by utilizing the mathematical model presented in Equation 5.1. Here, the Gaussian noise that ranged between the values 0 and 1 were applied to each datapoint in the dataset (i.e., addition to ascending dataset and subtraction to descending dataset).

5.2.4 Modeling

At this stage, the dataset was prepared and ready for modeling. The main goal of this stage was to develop the forecasting model that can make multi-step time series forecasting followed by a binary classification model that can classify failure instances.

As described earlier, the success of the proposed solution relies on the performance of the chosen prediction model (i.e., forecast and classification). This raised the emphasis on choosing the right model architecture besides the training hyperparameters. In this study, three RNN architectures were compared against each other in terms of accuracy in order to maintain the solution's highest performance.

The choice of RNN architecture was based on their efficiency in operating with sequential and time-series data as also motivated by [49]. The considered RNN architectures in the proposed solution were standard LSTM, CNN-LSTM, and GRU, where each model was developed and fine-tuned to make forecasts and classifications accordingly.

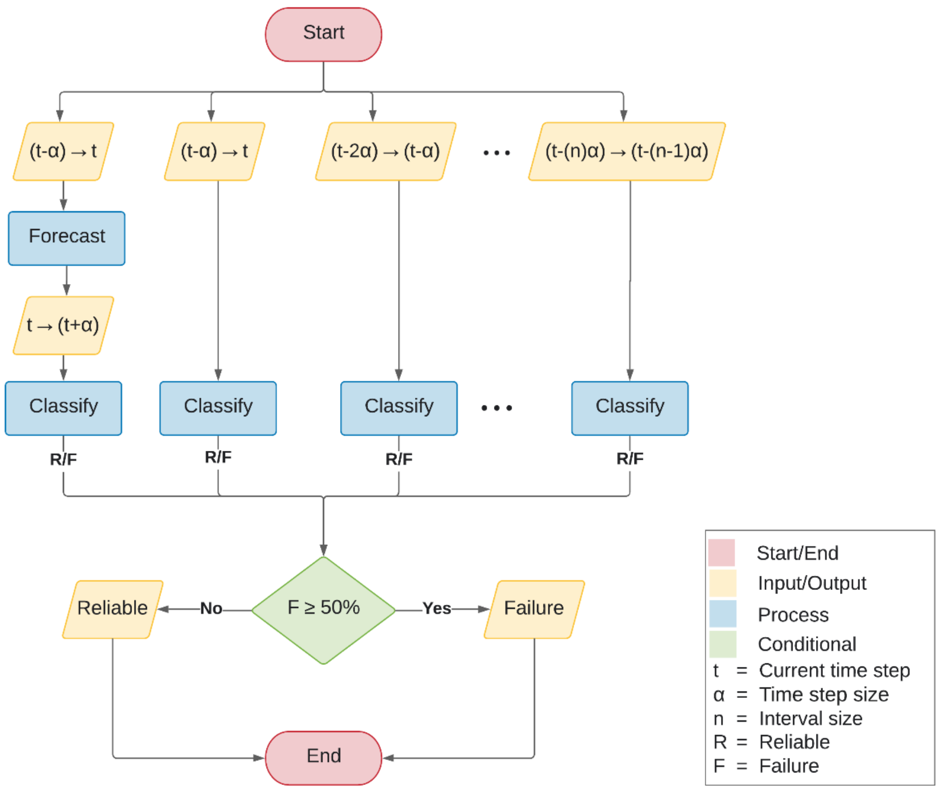


Figure 5.3, Flowchart for the proposed novel algorithm.

The proposed solution was developed to detect sensor drifts and thus predict failure. As illustrated in the flow chart in Figure 5.3 and the source code in Appendix A, the novel proposed algorithm, which will hereby be referred to as the *Predictive Sliding Detection Window* (PSDW), contains five main elements. Those elements are listed and described in the following list:

1- Current time step (t): The present hour in the observed time series.

2- Time step size (α): The size of each time interval.

3- Interval size (n): The total number of intervals in the sliding detection window.

4- Reliable label (R): Same or similar to ground truth sensor values.

5- Failure label (F): Abnormal sensor values.

The five elements in PSDW were facilitated by the developed forecasting and classification algorithms where their modeling can be described as follows:

The forecasting model was used to predict the temperature values for the specified interval size of hours. The dataset was segmented into adjustable interval sizes to make matching forecasted interval sizes. The total number of data points was adjusted to be divisible by the chosen interval size. Once the data is segmented, the dataset was separated into training and testing sets where the size of the training set was 80%, and the rest was for the testing set. In order to continuously improve the accuracy of the prediction model with a realistic approach. The model makes use of *walk-forward validation,* where it keeps track of all available time-series before making any further interval predictions.

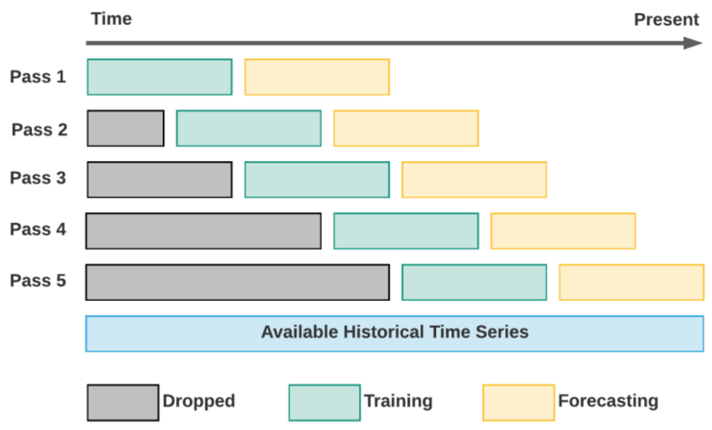


Figure 5.4, Walk-forward validation method in multistep-forecasting.

The mechanism of the walk-forward validation is demonstrated in Figure 5.4, where the available data points with a specified interval size are made available to the model. The model then is re-trained at the current time to make further accurate predictions. Furthermore, the size of data points that were available to the model was the same as the size of the forecasted interval (i.e., input interval size was equal to output interval size). This was made to easily keep track of the different forecasting algorithms as they were compared against each other. A comprehensive description on the evaluation method is presented in Section 5.3.

The classification model, on the other hand, was used to classify the observed values as reliable (i.e., true) or faulty (i.e., false). The three data sets (i.e., ground truth, ascending, and descending datasets) where initially divided into segments of one specific interval size chosen from the forecasting step (e.g., three, five, seven, nine, eleven). The chosen segmentation interval in this thesis was three for demonstration purposes. Since the three duplicated datasets were unlabeled and supervised techniques are of interest, it was required to label them systematically. Here, the ground truth data set was considered to be reliable and was therefore labeled as true. On the other side, the ascending and descending datasets were represented as faulty and were, therefore, label as false. Once the labeling procedure was completed, the all-segmented data points were shuffled to reduce any bias that might be formed in the classifier. The next step was carried out by training the model with the chosen RNN architectures (i.e., LSTM, CNN-LSTM, and GRU), followed by model evaluation as described in Section 5.2.5.

5.2.5 Model Evaluation

The evaluation phase is established once the building elements for training and evaluating the neural network are completed.

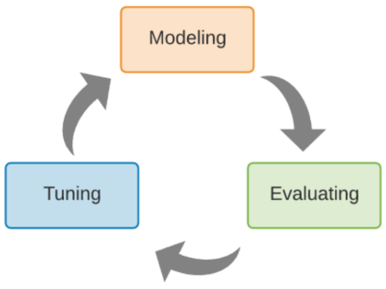


Figure 5.5, Model optimization procedure.

This phase consists of three main steps that ensure optimum performance of the built models. These steps are modeling, evaluating, and hyperparameter tuning and are illustrated in Figure 5.5.

The following sections will describe the chosen methodologies for evaluating the developed models, followed by a comprehensive description of the procedure of hyperparameter tuning. With regards to the forecasting task, the model's performance was evaluated using the Root Mean Squared Error (RMSE).

$$RMSE = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2} \tag{5.2}$$

As demonstrated in Equation 5.2, RMSE can be calculated by taking the square root of the squared differences between the actual value $y_j$and the forecasted value $\hat{y}_j$. This means that the magnitude of the error will drastically be penalized if more error was calculated.

In order to determine the most suitable model architecture (i.e., standard LSTM, CNN-LSTM, or GRU), the average RMSE for the time step size (α) was calculated to determine the performance of the respective forecasting model. The RMSE was also useful when comparing different models against each other as larger forecasting errors have a higher magnitude and thus could easily be noted.

In the case of the classification, the performance of each prediction model was evaluated by their F1-score and accuracy. The F1-score of the model was chosen as the metric for two main reasons. The first reason was because of the imbalance in the dataset where the number of true labels is not the same as the false labels. The second reason was due to high sensitivity to incorrectly classified cases.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5.3}$$

As Equation 5.3 demonstrates, the F1-score takes the precision and recall into account and calculates their harmonic mean. Similarly, the accuracy was evaluated to provide a clear overview of the number of correct predictions. Once the forecasting classification models were developed, they have subjected to hyperparameter tuning with predefined parameters and values. The goal was to find the most suited RMSE, F1-score, and accuracy for the respective forecasting and classification tasks. The predefined parameters in this stage were the interval size, number of layers, drop rate, batch size, number of epochs, activation functions, and the number of hidden units.

The hyperparameter search for both forecasting and classification models was optimized using a method called *hyperband.* This method was proposed as a research study and operated based on a random search but also has an adaptive resource allocation mechanism [50]. A detailed list of the chosen hyperparameters and their values is presented in Appendix B.

5.2.6 Deployment

Once the forecasting and classification models are trained with the optimum hyperparameters, they were deployed in AWS. This was made by uploading both models into a server application that ran on an AWS EC2 instance. The server was developed with the help of the python web framework *Flask* where each prediction model was initiated once a certain HTTP request was received at the endpoint. Here, the prediction was carried out using PSDW with CNN-LSTM models for forecasting and the classification, an interval size of one, and a multi-step prediction of three hours ahead.

5.3 Performance Evaluation Setup

Once a satisfying value of accuracy (i.e., RMSE, F1-score, and accuracy) was obtained, a sample of 10 predictions was made for each optimized algorithm (i.e., LSTM, CNN-LSTM, and GRU) and then statistically analyzed. The analysis procedure was carried out by calculating the SD, CI, ANOVA test, and Tukey's HSD test.

The standard deviation was used to determine the variation of the obtained prediction results. This includes all evaluation metrics involved in the prediction steps (i.e., RMSE, F1-score, and accuracy). The confidence interval of 95% was calculated to determine the validity of the observed mean values from the population. Subsequently, the overall differences between the forecasting and classification results were analyzed with the help of the one-way ANOVA test. Here, the significance of 0.05 was chosen to determine the overall in both forecasting and classification steps.

The forecasting step was analyzed in terms of the differences of RMSE values for interval size of each prediction model (i.e., LSTM, CNN-LSTM, and GRU) and RMSE overall variation between the three prediction models. In the classification step, the significance was analyzed to determine the overall differences in F1-score and accuracy for each prediction algorithm. Once all the overall significance was obtained, the measurements were then subjected to a Tukey's HSD test to determine the exact differences within the groups mentioned earlier (i.e., RMSE, F1-scores, and accuracy).

The impact of the proposed algorithm was then evaluated in the cloud computing environment EC2 in terms of response time ratio T. Here, the evaluation was carried out on a *t3. Medium EC2 instance* using a load testing framework *Locust*. The evaluation was designed to perform two different scenarios where both scenarios consisted of a range of one to a thousand simulated users. Each user was designed to iteratively make HTTP requests to the cloud computing entity at a rate of ~ 0.5 requests per second. In the first scenario, the users were responded with a prediction using the deployed algorithm. On the other hand, the second scenario was designed similarly to the first scenario but with only one difference as the users made empty HTTP requests.

$$Response\ time\ ratio\ (T)\ = \frac{T_p}{gcd\ (T_p, T_e)} : \frac{T_e}{gcd\ (T_p, T_e)} \qquad (5.4)$$

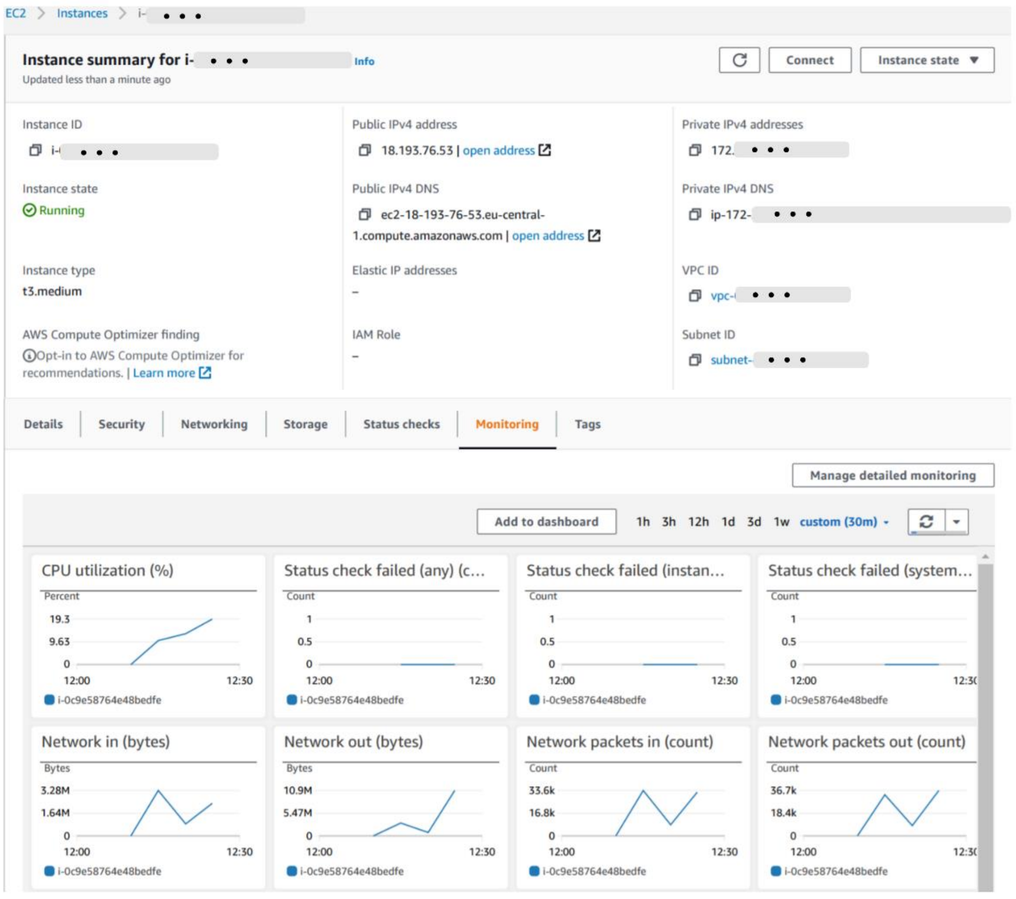
For the quantification of the impact of deploying the proposed algorithm, the ratio between the average response time from predictions T_p and empty request T_e was calculated. This ratio was obtained by utilizing Equation 5.4.

6 Results

This chapter presents all the results that were obtained from the experimental setup. The result introduces a screenshot of the developed system, data generation structure, followed by measurements from the forecasting, classification, and response time.

6.1 Resulting System

The resulting application consists of an EC2 instance that hosted the developed sensor fault detection model.



EC2 > Instances > i-[redacted]

Instance summary for i-[redacted] Info
Updated less than a minute ago

Connect | Instance state ▼

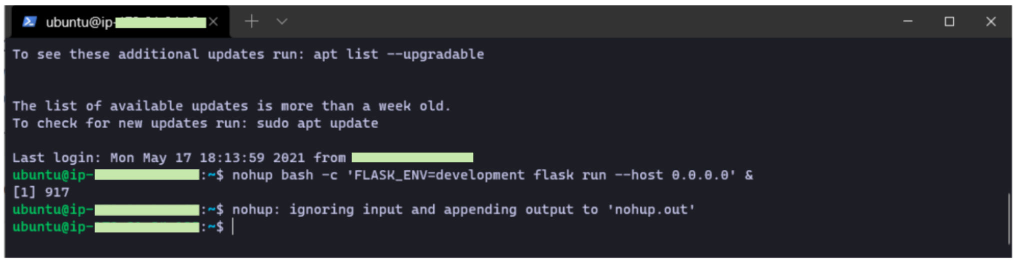
Instance ID	Public IPv4 address	Private IPv4 addresses
i-[redacted]	18.193.76.53 | open address	172.[redacted]
Instance state	**Public IPv4 DNS**	**Private IPv4 DNS**
Running	ec2-18-193-76-53.eu-central-1.compute.amazonaws.com | open address	ip-172-[redacted]
Instance type	**Elastic IP addresses**	**VPC ID**
t3.medium	–	vpc-[redacted]
AWS Compute Optimizer finding	**IAM Role**	**Subnet ID**
Opt-in to AWS Compute Optimizer for recommendations. | Learn more	–	subnet-[redacted]

Details | Security | Networking | Storage | Status checks | Monitoring | Tags

Manage detailed monitoring
Add to dashboard
1h 3h 12h 1d 3d 1w custom (30m)
CPU utilization (%)
Percent
19.3
9.63
0
12:00
12:30
i-0c9e58764e48bedfe
Status check failed (any) (c...
Count
1
0.5
0
Status check failed (instan...
Status check failed (system...
Network in (bytes)
Bytes
3.28M
1.64M
Network out (bytes)
10.9M
5.47M
Network packets in (count)
33.6k
16.8k
Network packets out (count)
36.7k
18.4k

Figure 6.1, Captured resulting application on AWS.

A running EC2 instance with a dashboard is presented in Figure 6.1.



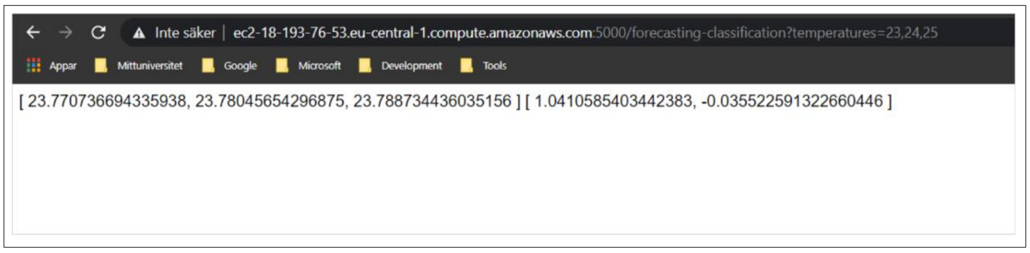
```
To see these additional updates run: apt list --upgradable


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Mon May 17 18:13:59 2021 from [redacted]
ubuntu@ip-[redacted]:~$ nohup bash -c 'FLASK_ENV=development flask run --host 0.0.0.0' &
[1] 917
ubuntu@ip-[redacted]:~$ nohup: ignoring input and appending output to 'nohup.out'
ubuntu@ip-[redacted]:~$ |
```

Figure 6.2, Command-line interface for the EC2 instance.

Figure 6.2 demonstrates the command line interface that was connected to the EC2 instance using a Secure Shell Protocol (SSH). The interface was used to run the server that hosted the prediction models.



Inte säker | ec2-18-193-76-53.eu-central-1.compute.amazonaws.com:5000/forecasting-classification?temperatures=23,24,25
Appar
Mittuniversitet
Google
Microsoft
Development
Tools

```
[ 23.770736694335938, 23.78045654296875, 23.788734436035156 ] [ 1.0410585403442383, -0.035522591322660446 ]
```

Figure 6.3, Resulting output from the proposed solution.

The output from the prediction models was displayed on the web browser, as demonstrated in Figure 6.3. The model took the sensor values as an input and output a forecasting result and a classification result. The three-step forecasting results are demonstrated in the right array while the binary classification at the left array.

6.2 Data Generation

The data points were formatted in JSON and consisted of four different attributes, namely the sensor type, temperature value in Celsius, humidity in percent, and the timestamp that consists of the date and time.

```
{
    "sensor_type": "hydroponic sensor",
    "temperature": "24.0",
    "humidity": "13.0",
    "datetime": "2021-03-10T04:01:59"
}
```

Figure 6.4, Sensor datapoint structure in JSON.

A sample of the collected data points is presented in Figure 6.4.

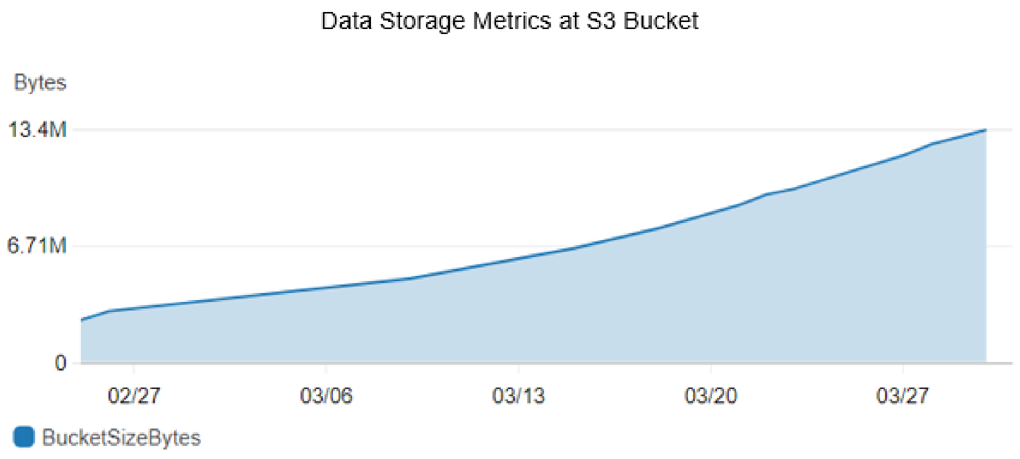


Figure 6.5, Storage of data progress at S3 Bucket.

The data collection progress at the storage entity in AWS is demonstrated in Figure 6.5. The plot from the S3 unit shows the number of accumulated data in bytes over the time between 26th of February and 31st of March.

6.3 Data Forecasting and Classification

Table 6.1, Evaluation of forecasting models based on RMSE and prediction interval size.

Forecasting Model	Interval Size	Avg. RMSE	SD RMSE	RMSE (±95% CI)
LSTM	3	0.5903	0.0012	0.0007
	5	0.6136	0.0014	0.0009
	7	0.6020	0.0053	0.0032
	9	0.6325	0.0186	0.0116
	11	0.6493	0.0173	0.0108
CNN-LSTM	3	0.5905	0.0014	0.0009
	5	0.6135	0.0027	0.0016
	7	0.5993	0.0069	0.0042
	9	0.6196	0.0084	0.0052
	11	0.6533	0.0344	0.0214
GRU	3	0.7408	0.1659	0.1030
	5	0.6818	0.0757	0.0469
	7	0.6198	0.0399	0.0248
	9	0.6035	0.0410	0.0254
	11	0.6174	0.0419	0.0260

The RMSE values of each RNN model with regards to the interval size are listed in Table 6.1. This includes statistical measurements of the mean, SD, and CI of 95% of the observations.

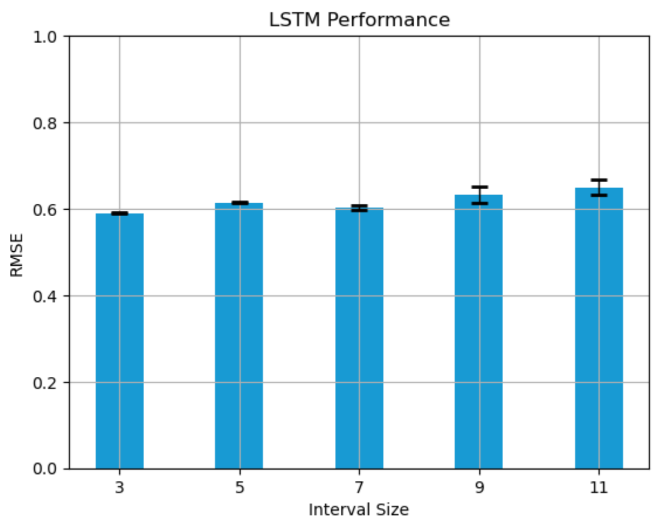


Figure 6.6, LSTM performance in RMSE concerning interval size.

The performance of the LSTM model in terms of the RMSE values of each interval size is illustrated in Figure 6.6. The lower RMSE value indicates a higher accuracy in prediction.

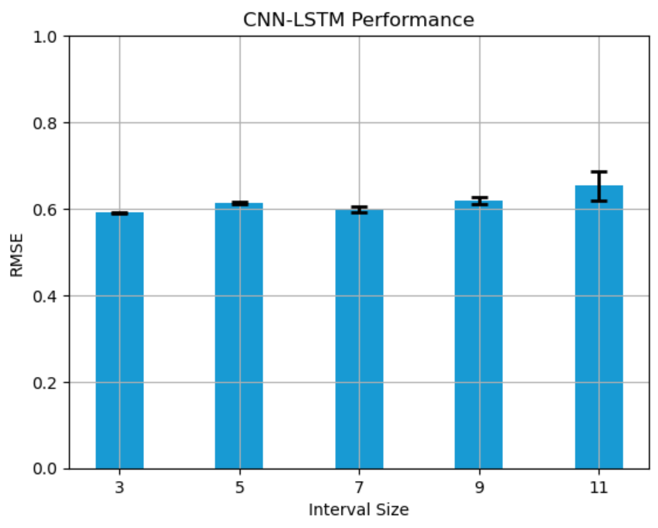


Figure 6.7, CNN-LSTM performance in RMSE concerning interval size.

The performance of the CNN-LSTM model in terms of the RMSE values of each interval size is illustrated in Figure 6.7. The lower RMSE value indicates a higher prediction accuracy.

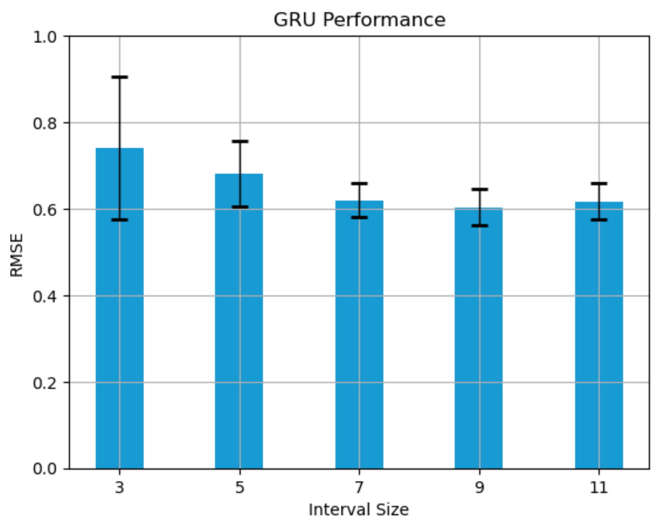
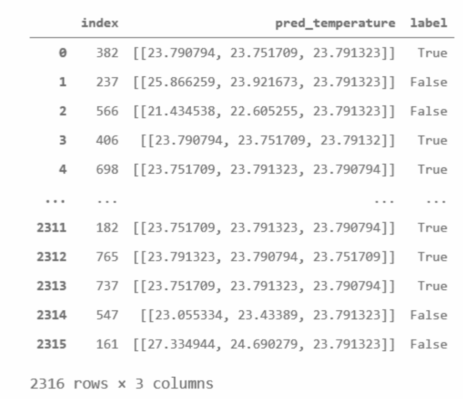


Figure 6.8, GRU performance in RMSE concerning interval size.

The performance of the GRU model in terms of the RMSE values of each interval size is illustrated in Figure 6.8. The lower RMSE value indicates a higher prediction accuracy.

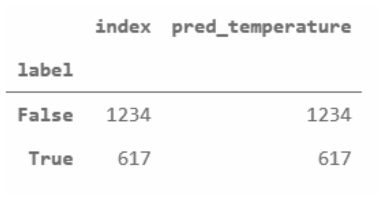


```
      index                        pred_temperature  label
0       382  [[23.790794, 23.751709, 23.791323]]   True
1       237  [[25.866259, 23.921673, 23.791323]]  False
2       566  [[21.434538, 22.605255, 23.791323]]  False
3       406   [[23.790794, 23.751709, 23.79132]]   True
4       698  [[23.751709, 23.791323, 23.790794]]   True
...     ...                                  ...    ...
2311    182  [[23.751709, 23.791323, 23.790794]]   True
2312    765  [[23.791323, 23.790794, 23.751709]]   True
2313    737  [[23.751709, 23.791323, 23.790794]]   True
2314    547   [[23.055334, 23.43389, 23.791323]]  False
2315    161  [[27.334944, 24.690279, 23.791323]]  False

2316 rows × 3 columns
```

Figure 6.9, Segmentation and labeled data points for classification.

The segmentation and labeling of the dataset are presented in Figure 6.9. The dataset contains three different datasets, the original dataset, ascending, and descending datasets with Gaussian noise. As demonstrated, the order of the data points was randomized and labeled in a segment with an interval size of three.



```
       index  pred_temperature
label
False   1234              1234
 True    617               617
```

Figure 6.10, An overview of the obtained datasets for classification.

An overview of the size of the dataset and assigned labels is illustrated in Figure 6.10.

Table 6.2, Evaluation of classification models based on F1-score.

Classification Model	Avg. F1 Score	SD F1-Score	F1-Score (±95% CI)
LSTM	0.7868	0.1588	0.0985
CNN-LSTM	0.7917	0.1669	0.1030
GRU	0.8167	0.1308	0.0811

The performance of the three classification models (i.e., LSTM, CNN-LSTM, and GRU) in terms of F1-score is listed in Table 6.2. The table also demonstrates the statistical mean, SD, and CI of 95% of the test sample.

Table 6.3, Evaluation of classification models based on accuracy.

Classification Model	Avg. Accuracy	SD Accuracy	Accuracy (±95% CI)
LSTM	0.7869	0.1575	0.0977
CNN-LSTM	0.8145	0.1710	0.1060
GRU	0.8244	0.1484	0.0920

The performance of the three classification models (i.e., LSTM, CNN-LSTM, and GRU) in terms of accuracy is listed in Table 6.3. The table also demonstrates the statistical mean, SD, and CI of 95% of the test sample.

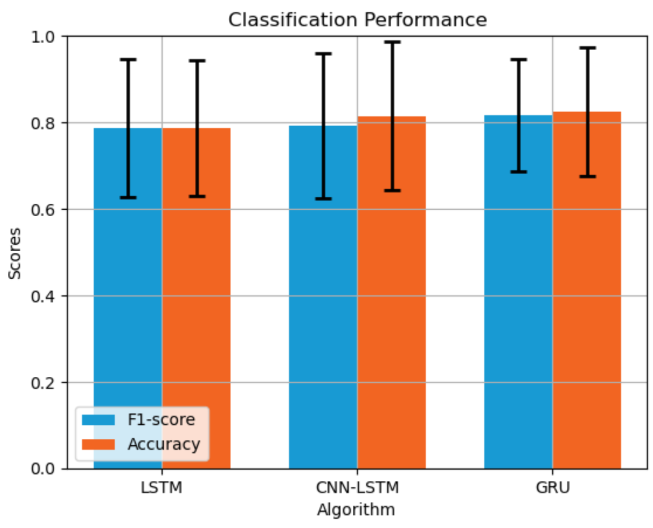


Figure 6.11, Classification performance in terms of F1-score and accuracy for interval size of three.

The performance of the classification models with regards to their F1-score and accuracy is demonstrated in Figure 6.11. The higher the score indicates better performance in predictions.

6.4 System Performance

The system's performance in terms of the average time to handle the HTTP requests is presented in this section.

Table 6.4, Performance comparison in terms of response time for EC2 instance with and without the proposed solution.

	Without PSDW	**With PSDW**
Total no. of users	1000	1000
Total no. of requests	215548	3290
Average Response Time (ms)	1358	77326
Min Response Time (ms)	59	1294
Max Response Time (ms)	4454	328457

An overview of the system performance with (and without) the deployed solution on the AWS EC2 instance is listed in Table 6.4. As demonstrated, the metrics of interest were the total number of the requesting users, the total number of HTTP requests, the mean, minimum, maximum - response time within the load test.

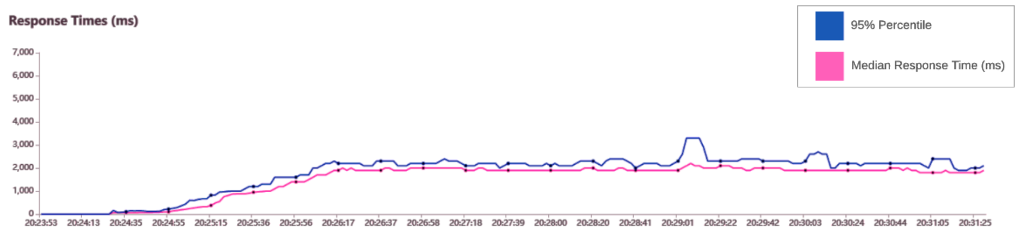


Figure 6.12, Median response time with 95% percentile of EC2 instance without the proposed solution.

Figure 6.12 illustrates the response time of the EC2 instance without the deployed prediction models. The duration of the load test was approx. eight minutes and the number of simulated users was accumulating with ten users until they reach a thousand. The pink plot illustrates the median response time in seconds, and the blue plot presents the values that lie below the median value of 95%.

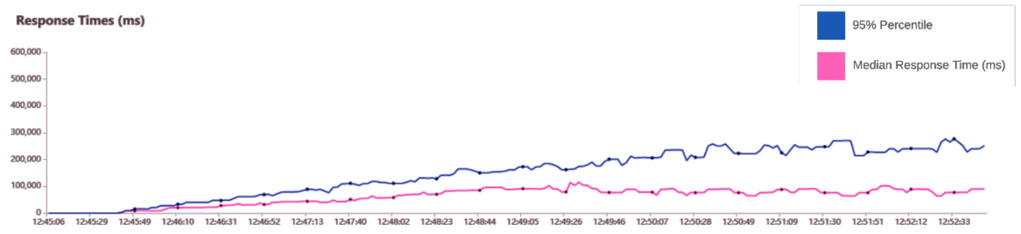


Figure 6.13, Median response time with 95% percentile of EC2 instance with the proposed solution.

The response time of the EC2 instance with the deployed prediction models during the load test is illustrated in Figure 6.13. Similar to the previous case, the duration of the load test was approx. eight minutes with a maximum number of a thousand for accumulating users. Here, the pink plot illustrates the median response time in seconds and the blue plot presents the values that lie below the median value of 95%. The response time ratio that was obtained from Equation 5.4 revealed a value of 38663: 679, which could be converted into 5694.1%

7 Discussion

This chapter provides a detailed analysis and discussion on the obtained results, chosen methodology, scientific implications, and contributions, followed by some ethical and societal considerations.

This thesis aimed to investigate and propose a novel predictive maintenance approach to detect sensor drift faults for smart hydroponic agriculture scenarios. The novel algorithm was given the name *Prediction Sliding Detection Window (PSDW)* and made use of recurrent neural networks to predict future sensor values and potential sensor drifting faults in data streams. The following sections provide a comprehensive analysis of each implemented and evaluated aspect that took part in this thesis (i.e., Data Generation, Data Forecasting and Classification, and System Performance). The results provided at the data generation layer will be discussed in terms of general observations, strengths, and limitations of the chosen methods.

The Data Forecasting and Classification layer will be comprehensively discussed in terms of statistical analysis of all significant findings, comprehensive interpretations and implications of the results, strengths, and limitations of the chosen approaches, and presents the scientific contributions. Lastly, the results provided by the system performance measurements will be discussed in terms of the feasibility of deploying the proposed solution in a cloud-based environment. Here, the impact of the overall response time will be quantified and analyzed.

7.1 Analysis of Data Generation Results

The data points that were produced by the sensor and the local server is illustrated in Figure 6.4. Each data point was structured in JSON format and contained four attributes: the sensor type, sensing temperature value, sensing humidity value, and timestamp in date/time format.

The proposed structure made simplified the readability of the sensing values and was convenient to work within the consequent steps. On the other hand, the data gathering procedure that was carried out using Amazon's Kinesis Firehose service is illustrated in Figure 6.5. As the graph revealed, the gathering procedure of the data in bytes tends to be non-linear.

There are two possible explanations that can describe this phenomenon. The first explanation could be related to the data compression method that is used by AWS, as the compression performance might vary according to the overall size of the data streams.

The second explanation could be due to the loss of some data points as they were dealt with in the data preparation stage in Section 5.2.3. As can be seen in Figure 6.5, the data gathering procedure took place between the 26th of February and 31st of March, and the dataset had a total size of ~ 13.5 Megabytes. This can be considered a limitation in this thesis as neural networks tend to perform better given bigger datasets.

7.2 Analysis of Data Forecasting and Classification Results

The discussion in this section will cover a statistical analysis of the forecasting and classification algorithms and a comprehensive interpretation of the results.

7.2.1 Forecasting

The forecasted values from the LSTM model tend to fit well with the actual temperature values as the mean RMSE values ranged between 0.5903 and 0.6493 based on the specified size of the interval (i.e., 3,5,7,9, and 11). The sample produced a low standard deviation and low margin of error (or confidence interval), suggesting that the obtained results are reliable and could represent the population. However, the results obtained from the ANOVA test suggest that there is an overall difference in the RMSE values. The observations from the interval sizes were then statistically analyzed pairwise using Tukey’s HSD test, as demonstrated in Appendix C. Here, the test revealed that the differences lied in pairs {(3, 5), (3, 9), (3, 11), (5, 9), (5, 11), (7, 9), (7, 11), and (9, 11)}, which could be visually identified in Figure 6.6.

Similarly, the forecasted values in the CNN-LSTM model provided a fitting value when evaluating the mean RMSE. Here, the values ranged between 0.5905 and 0.6533, which are very close to the LSTM model. The measured values have also revealed high reliability as the standard deviations, and the margin of errors was very low. In addition, the ANOVA test revealed an overall significant difference in the observed mean RMSE values of each interval size. Unlike LSTM, there were fewer RMSE values that expressed a significant difference, as also can be seen in Figure 6.7.

The Tukey's HSD test revealed that the significance could be found in the pairs of the intervals {(3, 5), (3, 9), (3, 11), (5, 11), (7, 11), (9, 11)} and could be further analyzed in Appendix C.

In contrast, the GRU model produced fewer fitting values when comparing the predicted values against the actual ones. Here, the mean RMSE values for each interval size ranged between 0.6035 and 0.7408 and, therefore, could be determined as the model with worse overall performance. As illustrated in Table 6.1 and Figure 6.8, a significantly higher standard deviation was obtained when the interval size was set to three. This value produced lower reliability in the quality of the measurements as the margin of error was slightly higher than the other developed model. On the other side, the ANOVA test showed that there is a significant overall difference between the RMSE values. The Tukey's HSD test showed that the differences lied in pairs {(3, 7), (3, 9), (3, 11)}. A more detailed analysis is demonstrated in Appendix C.

The one-on-one comparison between the developed models regarding respective interval sizes was analyzed and illustrated in Appendix C. Comparing the models at interval size three using the ANOVA and Tukey's HSD test showed a significant difference in the performance with regards to the mean RMSE values. More specifically, the differences occurred in pairs {(LSTM, GRU), (CNN-LSTM, GRU)}. The analysis of the interval size five showed the exact same results in terms of significant pairs, i.e., {(LSTM, GRU), (CNN-LSTM, GRU)}. The interval size of seven, nine, and eleven have shown no significant differences in mean RMSE values and, therefore could be considered as more stabilized values.

7.2.2 Classification

On the other hand, the performance of the classification models was demonstrated in Figure 6.11, Table 6.2, and Table 6.3. Here, the presented performance metrics show relatively similar F1 scores and accuracy values. In the case of the LSTM model, the model was able to reach a high mean F1-score and accuracy but with a relatively high standard deviation and margin of error.

This indicates a high alteration in the performance, which could lead to the unreliability of the classification model. Similarly, the CNN-LSTM model provided a high F1-score and accuracy but with the cost of a high standard deviation and margin of errors.

On the other side, the GRU model was able to achieve the highest F1-score and accuracy and with the lowest standard deviation and margin of error. However, the conducted statistical analysis using the ANOVA test revealed that there is no significant difference in performance in all three models, as illustrated in Appendix D.

The success of the classification step was achieved by utilizing the proposed labeling method. As illustrated in Figure 6.9, treating the dataset as segments with a matching size of the forecasting interval has simplified the labeling process. However, the choice of the interval size might be critical as the performance of the models was affected negatively when an interval size of three was chosen. This could be investigated in future researches as both hyperparameter optimization, and interval sizes should be considered. In general, the overall performance of the classification models was fair enough to be deployed for this particular scenario. This is due to the fact that false classifications don't necessarily impact the overall performance of the solution where errors are not penalized.

7.3 Analysis of System Performance Results

The response time that was provided by the EC2 instance ranged between 59 ms and 4454 ms at various workloads. This is considered to be normal according to an evaluation study conducted in [51]. On the other hand, utilizing the proposed solution in a cloud environment provided a range in response time between 1294 ms and 328457 ms. This means that the response time was significantly affected negatively by 5694.1%. One possible reason that could explain this tremendous delay is that the application is considered to be CPU intensive especially when increasing the number of users.

Another explanation could be caused by the server flask application that was deployed on the EC2 instance, as it was not designed to operate on scalable production environments. One possible solution to this phenomenon is to deploy the proposed solution in a more capable EC2 instance with better CPU and memory capabilities.

7.4 Method Discussion

This section provides a comprehensive discussion of each thesis milestone's chosen method (i.e., literature review, design, implementation, measure, and evaluation). This discussion covers the fitness of the chosen methods as well as their contribution to satisfying the thesis's overall aim.

Literature review: The chosen method for literature review has provided a broad understanding of well-recognized scientific resources in the fields of IoT, reliability theory, cloud computing, and ML. This method provided a comprehensive survey on the aforementioned fields and introduced three state-of-the-art RNN algorithms (i.e., LSTM, CNN-LSTM, and GRU) that were eventually used in this thesis. This milestone was ended with a general review of related works in the fields of predictive maintenance and sensor failure detection, where four scientific publications were reviewed and compared to this thesis. The comparison between this thesis and the reviewed articles has presented the significance of the proposed solution.

Design: The design of the proposed solution was influenced by the reviewed needs and requirements of indoor smart agriculture environments. These requirements were used to determine the essential components of a fitting solution (i.e., a cloud computing platform that utilizes ML algorithms for prediction). This also implied the necessity of developing a data pipeline to provide a proof-of-concept that can later be evaluated. It is worth noting that the proposed solution in this thesis only fits cases with similar problem statements and can therefore be generalized. This is due to the nature of deployed sensors that only take measurements in an indoor environment that tend to be stable and enduring, as mentioned earlier in Section 2.1.

Implement: The implementation milestone was structured in two different layers, namely data generation and data forecasting and classification layers. The data generation was carried out with a humidity and temperature sensor connected to a gateway deployed on an RPi. The RPi was connected to AWS IoT Hub service and was set up to transmit the sensory data per minute to then be stored in an S3 storage service. The setup used in the data generation stage was convenient to install as AWS provides services that ease the process of creating a secure data pipeline between different components.

However, there are some reservations when it comes to the creation of the dataset, as the number of collected data points was not enough for neural networks to process in the subsequent layers.

In the case of data forecasting and classification, the process of training, hyperparameter tuning, and evaluation was carried out on a local device. This was required to be economically effective, as each assigned computation at the cloud is considered a paid service. The process of model creation was structured by the CRISP-DM model. Here, the desired solution was formulated to fit the defined problems. However, there were a number of factors that affected the prediction results negatively in terms of accuracy measures. Those factors were the size of the dataset and the assigned time of hyperparameter tuning. In terms of the dataset, the number of data points was hardly enough for the neural networks to process for modeling. On the other hand, the time used to tune the hyperparameters was around thirty minutes for each neural network modeling. This was due to some limitations in the time and computational power that were available then.

Measure: The performance of the system and the prediction models were carried out in this milestone. The two main goals were to acquire system performance results and compare different neural networks' performance. In terms of the performance of the system, the average time to respond to forecasting and classification requests was taken. This was important due to the nature of the solution as it requires the computing entity to make continuous predictions to detect sensor drift faults. In terms of the prediction performance, the accuracy metrics results were acquired by estimating the RMSE values for the forecasting model as well as F1-score and accuracy for the classification model.

The RMSE was able to effectively determine the differences between the prediction multistep and the actual ones due to its heavy penalizing nature. In terms of the classification model, its performance was determined by calculating its F1-score and accuracy. The F1-score was useful for this particular scenario as the labels in the dataset tended to be imbalanced, and thus F1-score took into consideration unevenly distributed classes. In addition, F1-score was able to determine the tradeoff between the precision and the recall in any prediction attempt.

On the other hand, the accuracy was able to provide a result that could easily be interpreted as the number of correct classifications were analyzed against the total size of the dataset.

Evaluate: In this stage, all obtained results were statistically analyzed to gain a broader perspective on the observations in terms of variation of the values and significant differences. The analysis was carried out using the mean, SD, CI, ANOVA, and Tukey's HSD tests. The chosen methods provided a systematic technique for justifying the similarities and differences of the observation. This was useful to eliminate any uncertainties, especially when comparing similar values against each other.

7.5 Scientific Discussion

This section provides a comprehensive discussion on the scientific goals that this thesis covers as well as the contribution and encountered limitations. This thesis contributed to the field of predictive maintenance by proposing a novel algorithm that was later called PSDW. The algorithm was able to automate the detection of sensor drift faults by utilizing recurrent neural networks in predicting drifts in time series. More specifically, PSDW uses a forecasting model to make multi-step value predictions along with a classification model to detect drift faultiness in the sensor data streams, as described in Section 5.2.

On the other hand, due to limitations in time and resources, the proposed solution was only examined on a single temperature and humidity sensor. This limitation must be considered in future experiments as the proposed solution only works, given the assumptions mentioned in Section 5.2.2. This part was resolved in this thesis by utilizing a brand-new sensor that was assumed to be reliable. It is worth noting that the results provided by PSDW can only be generalized given the characteristics mentioned in Section 2.1.

The dataset in this thesis was subject to a labeling task in the classification stage. This thesis proposes a method for labeling the data for sensor drift detection. Here, an ascending and/or descending Gaussian noise was applied to two duplicates of the original dataset. All datasets were then segmented into a fixed interval size and labeled. The ascending/descending dataset was labeled as faulty.

This approach tends to be acceptable as the algorithm's detection window aims to detect overall trends in the data with an adjustable tolerance against faulty data points. Another approach that was considered for classifying was the utilization of RMSE values. With that regard, the obtained RMSE value resulting from the forecasting model could be used as north and southbound and all data points that lie in between could be considered reliable data.

This thesis also contributes to this field by proposing a benchmark comparison of three different RNN algorithms, namely LSTM, CNN-LSTM, and GRU. The analysis was based on comparing the performance (i.e., RMSE, F1-score, and Accuracy) of those algorithms in making time-series forecasting and classification for this particular scenario. The novel algorithm was deployed in a cloud computing environment to determine its performance in terms of response time. The results suggest that the chosen setup on the cloud computing environment was not optimum. One possible reason for that is the server setup, as it wasn't designed to handle a large number of requests. This could be further investigated in future researches, as stated in Section 8.3.

7.6 Ethical and Societal Considerations

Some ethical and societal aspects need to be considered in the context of predictive maintenance. This is due to the implications of such a maintenance scheme as adopting organizations tend to establish more automated and data-driven models. In the context of automation, there are some severe social considerations regarding the future of working places as automation could affect the number of work-forces and demand additional skillsets and knowledge base [52]. On the other side, the adoption of data-driven maintenance schemes (e.g., predictive maintenance) opens new possibilities to lower the environmental impact and increase the asset's life cycle. Adopting such schemes allows the businesses to make well-thought asset management, which provides many benefits such as reducing financial, environmental, and social costs [53].

8 Conclusions

The thesis was able to propose an automated drift detection method that could be used to increase the efficiency of maintenance. This proposed method consisted of a two-layered neural network that made use of forecasting and classification algorithms. The results from the developed models showed a reasonable amount of accuracy but with low reliability in classifying the faults. The proposed algorithm was later deployed in a cloud computing environment to test its operability in production environment. The results from the load tests shows a major trade-off in terms of response time. The results suggested the need of investigations for improving the accuracy and response time. This chapter briefly describes the thesis's milestones and scientific goals that contributed to fulfilling the overall aims. This includes a brief presentation on some possible future directions that could build upon this thesis.

8.1 Summary

In this thesis work, the fulfillment of the overall aim was carried out with the help of five main thesis milestone, where each goal was associated with a corresponding thesis milestone.

The first thesis milestone was to conduct a literature review to find the three most prominent solutions used in recurrent neural networks for time-series data streams. The chosen method was to conduct a comprehensive study on relevant fields (i.e., IoT, Reliability theory, Cloud computing, and ML) and find three state-of-the-art RNN algorithms (i.e., LSTM, CNN-LSTM, and GRU) that were eventually used in this thesis.

The second milestone was to design the experimental setup of the proposed prediction solution cloud computing environment. This was carried out by identifying the characteristics of the scenario, followed by proposing a data pipeline that manages the data streams in two different stages: data generation and data forecasting and classification. In addition, a proof-of-concept setup was designed to be deployed on a cloud computing environment to analyze its feasibility in terms of response time.

The third milestone concerns the implementation of the designed system setup (i.e., data generation, data forecasting and classification, and cloud system deployment). Here, the data produced using a temperature and humidity sensor and later collected and stored using amazon services. In addition, the data forecasting and classification were facilitated using the CRISP-DM model, as described in detail in Section 5.2.

The fourth milestone was to measure the performance of the developed system and ML models. The measures that were considered were the average response time for the cloud computing deployments and the different accuracy metrics for the prediction models, as explained comprehensively in Section 5.3.

Finally, the fifth milestone was to provide an analysis of the obtained results. This milestone was carried out using statistical analysis methods by evaluating the mean, SD, CI, ANOVA, and Tukey's HSD tests. This was useful to illuminate any uncertainties, especially when comparing similar values against each other.

8.2 Scientific Conclusions

This thesis aimed to investigate the possibilities of increasing the efficiency of traditional maintenance schemes. In particular, the thesis's main focus was to provide a solution that automates the detection of sensor drift faults using ML techniques. As an outcome, a novel algorithm was developed to make predictions on time series data streams using RNN algorithms. The algorithm was later named as Predictive Sliding Detection Window (PSDW) and consisted of forecasting and classification steps as described in details in Section 5.2 and illustrated in Figure 5.3.

This thesis contributes to this field by proposing a strategy for utilizing supervised learning techniques when dealing with time-series data streams. The strategy presents a segmentation method where the data streams' drifts were classified as faulty or reliable. The segmentation method proposed a high F1-score and accuracy but at the cost of a higher standard deviation. As discussed in Section 7.3, the thesis also presents an alternative strategy that could enhance the performance with more stability.

In addition, the thesis has also conducted a one-on-one comparison of three different recurrent neural networks (i.e., LSTM, CNN-LSTM, and GRU). The results were statistically analyzed as the significance in the performance metrics was evaluated using the one-way ANOVA test and Tukey's HSD test. The results suggested a slight similarity in the performance of the algorithms when dealing with the forecasting and classification tasks. More on the comparative analysis of the RNN models is presented in Section 7.1.2.

The proposed novel algorithm was deployed in a cloud computing environment and evaluated in terms of response time. The evaluation revealed that the response time was significantly affected during load testing. As suggested in Section 7.1.3, the results from the deployment of the proposed solution in a cloud computing environment indicate the need for further investigation as described in Section 8.3.2.

Finally, predictive maintenance that enhances smart agriculture applications has proven to provide a vast potential in managing the deployed assets. As this thesis has presented, the deployed sensors can be monitored using machine learning approaches and techniques. The generated data could be enhanced by machine learning that could significantly empower the future of IoT.

8.3 Future Work

There is a number of directions that can be considered as a part of future work. These directions might be related to improving the performance of the prediction models and investigation with different hosting systems. The following sections briefly describe two of the most prominent approaches that can be taken as a next research step.

8.3.1 Prediction Performance Improvements

Providing different approaches for obtaining better accuracy is the main focus of this research direction. One prominent approach for achieving that is the use of ensemble learning, where better accuracy is obtained from a combination of ML models [54]. Here, the proposed PSDW will use different ML algorithms to forecast new time series and classify them as faulty or reliable. Furthermore, the proposed PSDW algorithm can also be enhanced with a multi-classifier to identify different faults and thus help perform root-cause-analysis.

8.3.2 Computing Paradigms Evaluation

It would be beneficial to conduct more research on the different system architectures, specifically by analyzing the proposed solution's performance tradeoffs on the edge and low-powered IoT devices. Here, metrics such as processing power and energy consumption would be of interest. This is due to the nature of the drift detection mechanism, as continuous analysis of the data streams is required.

References

[1] L. Atzori, A. Iera, and G. Morabito, 'The Internet of Things: A survey', Comput. Networks, vol. 54, no. 15, pp. 2787–2805, Oct. 2010, doi: 10.1016/j.comnet.2010.05.010

[2] T. Zonta, C. A. da Costa, R. da Rosa Righi, M. J. de Lima, E. S. da Trindade, and G. P. Li, 'Predictive maintenance in the Industry 4.0: A systematic literature review', Comput. Ind. Eng., vol. 150, p. 106889, Dec. 2020, doi: 10.1016/j.cie.2020.106889.

[3] V. Palande, A. Zaheer, and K. George, 'Fully Automated Hydroponic System for Indoor Plant Growth', in *Procedia Computer Science*, Jan. 2018, vol. 129, pp. 482–488, doi: 10.1016/j.procs.2018.03.028.

[4] D. Karimanzira and T. Rauschenbach, 'Enhancing aquaponics management with IoT-based Predictive Analytics for efficient information utilization', Inf. Process. Agric., vol. 6, no. 3, pp. 375–385, Sep. 2019, doi: 10.1016/j.inpa.2018.12.003.

[5] S. Madakam, R. Ramaswamy, and S. Tripathi, 'Internet of Things (IoT): A Literature Review', *J. Comput. Commun.*, vol. 03, no. 05, pp. 164–173, 2015, doi: 10.4236/jcc.2015.35021.

[6] E. Navarro, N. Costa, and A. Pereira, 'A systematic review of iot solutions for smart farming', *Sensors (Switzerland)*, vol. 20, no. 15. MDPI AG, pp. 1–29, Aug. 01, 2020, doi: 10.3390/s20154231.

[7] O. Elijah, T. A. Rahman, I. Orikumhi, C. Y. Leow and M. N. Hindia, "An Overview of Internet of Things (IoT) and Data Analytics in Agriculture: Benefits and Challenges," in IEEE Internet of Things Journal, vol. 5, no. 5, pp. 3758-3773, Oct. 2018, doi: 10.1109/JIOT.2018.2844296.

[8] A. Gaddam, T. Wilkin, M. Angelova, and J. Gaddam, 'Detecting Sensor Faults, Anomalies and Outliers in the Internet of Things: A Survey on the Challenges and Solutions', *Electronics*, vol. 9, no. 3, p. 511, Mar. 2020, doi: 10.3390/electronics9030511.

[9] A. B. Sharma, L. Golubchik, and R. Govindan, 'Sensor faults: Detection methods and prevalence in real-world datasets', *ACM Trans. Sens. Networks*, vol. 6, no. 3, pp. 1–39, Jun. 2010, doi: 10.1145/1754414.1754419.

[10] M. Takruri, S. Challa, and R. Chakravorty, 'Recursive Bayesian Approaches for Auto Calibration in Drift Aware Wireless Sensor Networks', 2010, doi: 10.4304/jnw.5.7.823-832.

[11] P. P. Ray, 'Internet of things for smart agriculture: Technologies, practices and future direction', *J. Ambient Intell. Smart Environ.*, vol. 9, no. 4, pp. 395–420, Jan. 2017, doi: 10.3233/AIS-170440.

[12] R. Lakshmanan, M. Djama, S. K. Selvaperumal, and R. Abdulla, 'Automated smart hydroponics system using internet of things', *Int. J. Electr. Comput. Eng.*, vol. 10, no. 6, pp. 6389–6398, 2020, doi: 10.11591/ijece.v10i6.pp6389-6398.

[13] G. Marques, D. Aleixo, and R. Pitarma, 'Enhanced Hydroponic Agriculture Environmental Monitoring: An Internet of Things Approach', in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Jun. 2019, vol. 11538 LNCS, pp. 658–669, doi: 10.1007/978-3-030-22744-9_51.

[14] T. Gomiero, 'Food quality assessment in organic vs. conventional agricultural produce: Findings and issues', *Applied Soil Ecology*, vol. 123. Elsevier B.V., pp. 714–728, Feb. 01, 2018, doi: 10.1016/j.apsoil.2017.10.014.

[15] D. Bandyopadhyay and J. Sen, "Internet of Things: Applications and Challenges in Technology and Standardization," *Wirel. Pers Commun*, vol. 58, pp. 49–69, 2011, doi: 10.1007/s11277-011-0288-5.

[16] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," in IEEE Communications Surveys & Tutorials, vol. 17, no. 4, pp. 2347-2376, Fourthquarter 2015, doi: 10.1109/COMST.2015.2444095.

[17] M. Modarres, M. Kaminskiy and V. Krivtsov, Reliability engineering and risk analysis, 3rd ed. CRC Press, 2017, pp. 75-80.

[18] A. Birolini, Reliability Engineering: Theory and Practice, 8th ed. Springer-Verlag Berlin Heidelberg, 2017, pp. 19-20.

[19] E. Dubrova, B. / Dordrecht, and / London, 'FAULT TOLERANT DESIGN: AN INTRODUCTION'.

[20] M. Armbrust et al., 'A view of cloud computing', Communications of the ACM, vol. 53, no. 4. pp. 50–58, Apr. 01, 2010, doi: 10.1145/1721654.1721672.

[21] Q. Zhang, L. Cheng, and R. Boutaba, 'Cloud computing: state-of-the-art and research challenges', *J Internet Serv Appl*, vol. 1, pp. 7–18, 2010, doi: 10.1007/s13174-010-0007-6.

[22] M. Mohri, A. Rostamizadeh and A. Talwalkar, Foundations of Machine Learning, 2nd ed. MIT Press, 2018, pp. 1-3.

[23] C. Aggarwal, Neural networks and deep learning, 1st ed. New York: Springer, 2019.

[24] S. Haykin, Neural Networks and Learning Machines, 3rd ed. New York, N.Y.: Pearson Education Inc., 2009, pp. 10-12.

[25] R. Yamashita, M. Nishio, R. Kinh, G. Do, and K. Togashi, 'Convolutional neural networks: an overview and application in radiology', doi: 10.1007/s13244-018-0639-9.

[26] X. Zhang, Y. Wang, N. Zhang, D. Xu, and B. Chen, 'Research on Scene Classification Method of High-Resolution Remote Sensing Images Based on RFPNet', *Appl. Sci.*, vol. 9, no. 10, p. 2028, May 2019, doi: 10.3390/app9102028.

[27] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink and J. Schmidhuber, "LSTM: A Search Space Odyssey," in IEEE Transactions on Neural Networks and Learning Systems, vol. 28, no. 10, pp. 2222-2232, Oct. 2017, doi: 10.1109/TNNLS.2016.2582924.

[28] J. Chung, C. Gulcehre, and K. Cho, 'Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling'.

[29] C. Lin, D. Deng, C. Kuo and L. Chen, "Concept Drift Detection and Adaption in Big Imbalance Industrial IoT Data Using an Ensemble Learning Method of Offline Classifiers," in IEEE Access, vol. 7, pp. 56198-56207, 2019, doi: 10.1109/ACCESS.2019.2912631.

[30] J. C. P. Cheng, W. Chen, K. Chen, and Q. Wang, 'Data-driven predictive maintenance planning framework for MEP components based on BIM and IoT using machine learning algorithms', Autom. Constr., vol. 112, p. 103087, Apr. 2020, doi: 10.1016/j.autcon.2020.103087.

[31] H. Liu, M. Huang, I. Janghorban, P. Ghorbannezhad and C. Yoo, "Faulty sensor detection, identification and reconstruction of indoor air quality measurements in a subway station," 2011 11th International Conference on Control, Automation and Systems, 2011, pp. 323-328.

[32] D. Kumar, S. Rajasegarar and M. Palaniswami, "Automatic Sensor Drift Detection and Correction Using Spatial Kriging and Kalman Filtering," 2013 IEEE International Conference on Distributed Computing in Sensor Systems, 2013, pp. 183-190, doi: 10.1109/DCOSS.2013.52.

[33] 'What is AWS'. https://aws.amazon.com/what-is-aws/?nc1=f_cc (accessed Apr. 29, 2021).

[34] 'Compute Services - Overview of Amazon Web Services'. https://docs.aws.amazon.com/whitepapers/latest/aws-overview/compute-services.html (accessed Apr. 29, 2021).

[35] 'Internet of Things (IoT) - Overview of Amazon Web Services'. https://docs.aws.amazon.com/whitepapers/latest/aws-overview/storage-services.html (accessed Apr. 29, 2021).

[36] 'Security, Identity, and Compliance - Overview of Amazon Web Services'. https://docs.aws.amazon.com/whitepapers/latest/aws-overview/security-services.html (accessed Apr. 29, 2021).

[37] 'Internet of Things (IoT) - Overview of Amazon Web Services'. https://docs.aws.amazon.com/whitepapers/latest/aws-overview/internet-of-things-services.html (accessed Apr. 29, 2021).

[38] 'Machine Learning - Overview of Amazon Web Services'. https://docs.aws.amazon.com/whitepapers/latest/aws-overview/machine-learning.html (accessed Apr. 29, 2021).

[39] 'Get to Know Azure | Microsoft Azure'. https://azure.microsoft.com/en-us/overview/ (accessed Apr. 30, 2021).

[40] 'Azure Compute—Virtualization and Scalability | Microsoft Azure'. https://azure.microsoft.com/en-us/product-categories/compute/ (accessed Apr. 30, 2021).

[41] 'Overview of Linux VMs in Azure - Azure Virtual Machines | Microsoft Docs'. https://docs.microsoft.com/en-us/azure/virtual-machines/linux/overview (accessed Apr. 30, 2021).

[42] 'Azure virtual machine scale sets overview - Azure Virtual Machine Scale Sets | Microsoft Docs'. https://docs.microsoft.com/en-us/azure/virtual-machine-scale-sets/overview (accessed Apr. 30, 2021).

[43] 'Azure Functions Overview | Microsoft Docs'. https://docs.microsoft.com/en-us/azure/azure-functions/functions-overview (accessed Apr. 30, 2021).

[44] 'Introduction to Blob (object) storage - Azure Storage | Microsoft Docs'. https://docs.microsoft.com/en-us/azure/storage/blobs/storage-blobs-introduction (accessed Apr. 30, 2021).

[45] 'Azure Security Services and Technologies | Microsoft Docs'. https://docs.microsoft.com/en-us/azure/security/fundamentals/services-technologies (accessed Apr. 30, 2021).

[46] 'Introduction to Azure IoT Hub | Microsoft Docs'. https://docs.microsoft.com/en-us/azure/iot-hub/about-iot-hub (accessed Apr. 30, 2021).

[47] ‘What is Azure Machine Learning | Microsoft Docs’. https://docs.microsoft.com/en-us/azure/machine-learning/overview-what-is-azure-ml (accessed Apr. 30, 2021).

[48] B. S. Đorđević, S. P. Jovanović and V. V. Timčenko, "Cloud Computing in Amazon and Microsoft Azure platforms: Performance and service comparison," 2014 22nd Telecommunications Forum Telfor (TELFOR), 2014, pp. 931-934, doi: 10.1109/TELFOR.2014.7034558.

[49] Y. Bengio, P. Simard, and P. Frasconi, ‘Learning Long-Term Dependencies with Gradient Descent is Difficult’, *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994, doi: 10.1109/72.279181.

[50] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, ‘Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization’, *J. Mach. Learn. Res.*, vol. 18, pp. 1–52, Mar. 2016, Accessed: May 02, 2021. [Online]. Available: http://arxiv.org/abs/1603.06560.

[51] J. Dejun, G. Pierre, and C. H. Chi, ‘EC2 performance analysis for resource provisioning of service-oriented applications’, in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Nov. 2010, vol. 6275 LNCS, pp. 197–207, doi: 10.1007/978-3-642-16132-2_19.

[52] M. Muro, R. Maxim, and J. Whiton, ‘Automation and Artificial Intelligence: How Machines are Affecting People and Places’, Brookings India, Jan. 2019. Accessed: May 04, 2021. [Online]. Available: http://hdl.handle.net/11540/9686.

[53] P. Brous, M. Janssen, D. Schraven, J. Spiegeler, and B. Can Duzgun, ‘Factors Influencing Adoption of IoT for Data-driven Decision Making in Asset Management Organizations’, 2017, doi: 10.5220/0006296300700079.

[54] O. Sagi and L. Rokach, ‘Ensemble learning: A survey’, *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.*, vol. 8, no. 4, p. e1249, Jul. 2018, doi: 10.1002/widm.1249.

Appendix A: Source Code

https://github.com/Ayads/DT005A

Appendix B: Hyperparameter Setup

Forecasting Model	**Parameter**	**Search Space**	**Selected Value**	**Mean RMSE**
LSTM	Interval Size	3, 5, 7, 8, 9, 11	3	0.5903
	No. of layers	1, 3, 5, 7	5	
	Drop rate	0.0, 0.2, 0.4, 0.6, 0.8	0.4	
	Batch sizes	32, 64, 128, 256, 512	32	
	No. of epochs	1	10	
	Activation function	relu, tanh, sigmoid	tanh	
	Hidden units	50, 100, 150, 200	200	
CNN LSTM	Interval Size	3, 5, 7, 8, 9, 11	3	0.5905
	No. of layers	1, 3, 5, 7	7	
	Drop rate	0.0, 0.2, 0.4, 0.6, 0.8	0.2	
	Batch sizes	32, 64, 128, 256, 512	0	
	No. of epochs	1	10	
	Activation function	relu, tanh, sigmoid	tanh	
	Hidden units	50, 100, 150, 200	200	
GRU	Interval Size	3, 5, 7, 8, 9, 11	7	0.6198
	No. of layers	1, 3, 5, 7	3	
	Drop rate	0.0, 0.2, 0.4, 0.6, 0.8	0.2	
	Batch sizes	32, 64, 128, 256, 512	32	
	No. of epochs	1	10	
	Activation function	relu, tanh, sigmoid	relu	
	Hidden units	50, 100, 150, 200	150	

Classification Model	**Parameter**	**Search Space**	**Selected Value**	**Mean F1-Score**	**Mean Accuracy**
LSTM	No. of layers	1, 3, 5, 7	1	0.7868	0.7869
	Drop rate	0.0, 0.2, 0.4, 0.6, 0.8	0		

	Batch sizes	32, 64, 128, 256, 512	32		
	Max no. of epochs	1 to 50	50		
	Activation function	relu, tanh, sigmoid	relu		
	Hidden units	50, 100, 150, 200	200		
CNN LSTM	No. of layers	1, 3, 5, 7	1	0.7917	0.8145
	Drop rate	0.0, 0.2, 0.4, 0.6, 0.8	0		
	Batch sizes	32, 64, 128, 256, 512	32		
	Max no. of epochs	1	50		
	Activation function	relu, tanh, sigmoid	sigmoid		
	Hidden units	50, 100, 150, 200	200		
GRU	No. of layers	1, 3, 5, 7	1	0.8167	0.8244
	Drop rate	0.0, 0.2, 0.4, 0.6, 0.8	0		
	Batch sizes	32, 64, 128, 256, 512	32		
	Max no. of epochs	1	50		
	Activation function	relu, tanh, sigmoid	tanh		
	Hidden units	50, 100, 150, 200	100		

Appendix C: Forecasting Tukey's HSD Analysis

Algorithm	**Interval Size Pair**	**Q.05 = 4.0184 Q.01 = 4.8927**	**Significant Difference**
LSTM	3: 5	Q = 5.98 (p = .00103)	Yes
	3: 7	Q = 3.00 (p = .22801)	No
	3: 9	Q = 10.81 (p = .00000)	Yes
	3: 11	Q = 15.15 (p = .00000)	Yes
	5: 7	Q = 2.98 (p = .23551)	No
	5: 9	Q = 4.83 (p = .01138)	Yes
	5: 11	Q = 9.17 (p = .00000)	Yes
	7: 9	Q = 7.80 (p = .00002)	Yes
	7: 11	Q = 12.14 (p = .00000)	Yes
	9: 11	Q = 4.34 (p = .02850)	Yes
CNN-LSTM	3: 5	Q = 4.27 (p = .03225)	Yes
	3: 7	Q = 1.63 (p = .77900)	No
	3: 9	Q = 5.38 (p = .00373)	Yes
	3: 11	Q = 11.61 (p = .00000)	Yes
	5: 7	Q = 2.64 (p = .34841)	No
	5: 9	Q = 1.11 (p = .93387)	No
	5: 11	Q = 7.34 (p = .00005)	Yes
	7: 9	Q = 3.75 (p = .07756)	No
	7: 11	Q = 9.98 (p = .00000)	Yes

	9: 11	Q = 6.23 (p = .00059)	Yes
GRU	3: 5	Q = 2.02 (p = .61201)	No
	3: 7	Q = 4.14 (p = .04028)	Yes
	3: 9	Q = 4.71 (p = .01437)	Yes
	3: 11	Q = 4.23 (p = .03485)	Yes
	5: 7	Q = 2.12 (p = .56768)	No
	5: 9	Q = 2.68 (p = .33332)	No
	5: 11	Q = 2.20 (p = .53108)	No
	7: 9	Q = 0.56 (p = .99455)	No
	7: 11	Q = 0.08 (p = .00000)	No
	9: 11	Q = 0.48 (p = .99704)	No

Interval Size / **Algorithm Pair**	**3**	**5**	**7**	**9**	**11**
LSTM: CNN-LSTM	No	No	No	No	No
LSTM: GRU	Yes	Yes	No	No	No
CNN-LSTM: GRU	Yes	Yes	No	No	No
	Significant Difference at Q.05 = 3.5064 Q.01 = 4.4948				

Appendix D: Classification Tukey's HSD Analysis

Performance Metric / Algorithm Pair	F1-Score	Accuracy
LSTM: CNN-LSTM	No	No
LSTM: GRU	No	No
CNN-LSTM: GRU	No	No
	Significant Difference at Q.05 = 3.5064 Q.01 = 4.4948	