University of Bremen

Faculty - FB01

Demonstration machine learning for LoRa Sensors and integration into a streaming platform

Master Project in Communication and Information Technology (CIT)

Submitted by: Md Selim Hossain
Matriculation No: 3202203

Examiner: Dr.-Ing. Reiner Jedermann

Supervisor: Dr.-Ing. Reiner Jedermann

IMSAS
Institut für Mikrosensoren, -aktoren und -systeme

Declaration of Independence

I hereby confirm that this study was completed entirely by me, with the exception of official support from the Institute for Micro sensors, -actuators and –systems (IMSAS), at the University of Bremen in Germany. I have written this project independently and have not used any resources other thanthose specified. The part of the work, which is taken from the wording or the meaning of other works, hasbeen identified and the source has been indicated.

Abstract

Sensors or End-Nodes, a Gateway, and a Network/Cloud-Server make up a conventional LoRaWAN Network. The data which is in Jason format for each file have been extracted from the hard disk and the data have been cleaned for the further process The sensor data have been converted to HEX format for transmission. MQTT (Message Queuing Telemetry Transport) platform has been used in this project for data transmission. Data transmitted by a RN2483 node is received by multiple LoRa gateways, which forward the received data packets to a centralized network server (The Things Network). For the next step, the sensor data have been subscribed to and decoded. Afterwards, receiving all the data, the pre-exponential factor, rate constant, bias, mse, and average have been calculated by using curve fitting algorithm. Different machine learning models (Logistic regression, Support Vector Machine, and Neural Networks) have been used to determine whether the file is in good or bad cooling.

Table of Contents

Declaration of Independence
Abstract
List of Figures
List of table
1 Introduction 1
2 Overview of LoRa Network Architecture and Elements 2
2.1 LoRaWAN Classes 3
2.2 Class-A 3
2.3 Class-B 3
2.4 Class-C 3
2.5 Gateway 3
2.6 MQTT Access to TTN Server 4
2.7 Maximum packet size 4
2.8 Frame Structure 5
3 Evaluation of Event Driven Architecture 7
3.1 MQTT as Event-Driven Architecture 10
3.1.1 Architecture 11
3.1.2 Advantages 13
3.1.3 Disadvantages 14
3.1.4 Applications 14
4 System design and installation 16
4.1 Components and Data Flow 17
4.1.1 Data Source 17
4.1.2 Raspberry Pi and Display 18
4.1.3 LoRaWAN Gateway 18
4.1.4 MQTT Broker 19
4.1.5 Models 20
4.2 Installation overview 20
4.2.1 Installation of MQTT broker 20
5 Curve Fitting 21
5.1 Categories 21
5.2 Least-Squares Algorithm 23

5.3 Exponential Curve Fitting 23
5.4 Curve Fitting Python API 24
5.5 Pros and Cons of Curve Fitting 30
6 Machine Learning Model 31
6.1 Logistic regression 31
6.1.1 Types of logistic regression 32
6.1.2 Advantages and Disadvantages of using logistic regression 32
6.2 Logistic regression using scikit-learn 32
6.3 Support Vector Machine 33
6.3.1 Types of SVM 34
6.3.2 Hyperplane and Support Vectors in the SVM algorithm 35
6.3.3 Pros and Cons associated with SVM 35
6.4 Implementation of support vector machines in Scikit-Learn 35
7 Methods 37
7.1 Comparison of different Mathematical Model 37
8 Results 39
8.1 Evaluation of Logistic Regression: 39
8.2 Evaluation of Support Vector Machine: 41
8.3 Evaluation of Neural Network Model: 42
9 Problems During Implementation 45
10 Conclusion 47
11 Future research 47
12 Appendix A 48
12.1 Subscribe the data from TTN server 48
12.2 Publish the data to TTN Server 48
12.3 Logistic Regression Model 48
12.4 Necessary Python Library for the Project 49
13 Bibliography 50

List of figures

Figure 2.1 LoRaWAN Architecture 2
Figure 2.2 LoRa Frame Format 5
Figure 3.1 Architecture of Event Driven Platform 7
Figure 3.2Comparison of Publish-Subscribe mechanism with Point-to-Point Communication 9
Figure 3.3 MQTT Architecture 11
Figure 3.4 MQTT Publisher and Subscriber 12
Figure 3.5 Example of Humidity sensor with Publish and Subscriber 13
Figure 4.1 Software Architecture 16
Figure 4.2 RN2483 Sensors 17
Figure 4.3 Raspberry Pi and Display 18
Figure 4.4 LoRa Gateway 19
Figure 5.1 Category of Curve Fitting 22
Figure 5.2 Exact Fit Curve Fitting 22
Figure 5.3 Original data vs Curve fitting data 25
Figure 5.4 Pair Plot of the Parameters 28
Figure 5.5 Correlation between the Parameters 29
Figure 6.1 Logistic regression applied to a range of −20 to 20 31
Figure 6.2 Support vector machine for classification illustration 34
Figure 7.1 Confusion matrix 38
Figure 7.2 Confusion matrix of testing set and its heat map 40
Figure 7.3 Logistic Regression Classification report 40
Figure 7.4 Confusion matrix of testing set and its heat map 41
Figure 7.5 Support Vector Machine Classification report. 42
Figure 7.6 Confusion matrix of testing set and its heat map. 43
Figure 7.7 . Neural Network Classification report for 500 epochs. 43
Figure 7.8 Confusion matrix of testing set and its heat map 44
Figure 7.9 Neural Network Classification report for 2000 epochs. 44
Figure 7.10 Deep Neural Network losses 45

List of tables

Table 1 Parameters of the Dataset..27

1 Introduction

The term "Long Range" (or LoRa for short) refers to a patented wireless data communication method. Cycleo of Grenoble, France invented the technology, which Semtech purchased in 2012. The main characteristic that sets LoRa apart from other other wireless WAN technologies is its ability to transmit over vast distances while using little power. The support for low bit rate applications, however, places less of a burden on mobility and dependability. This technology is being developed to support Low Power Wide Area Networks (LPWANs), a class of wireless wide area network that aims to outperform legacy cellular systems by +20 dB [1].

The work mainly included:

- Sending data in a block to increase the efficiency.
- Data transmission to the end devices via over-the-air activation (OTAA).
- Using serial ports to connect end devices allows you to read temperature and humidity sensor data instantly.
- Working with Python Programming to connect the end devices (Sensors) with the Gateway.
- Making use of the Message Queuing Telemetry Transport (MQTT) protocol to send data to MQTT streaming infrastructure.
- Calculate different parameters from the datasets using Curve fitting algorithm.
- Constructing various machine learning models using the datasets.
- Classify the testing dataset from the Model.

•

2 Overview of LoRa Network Architecture and Elements

A typical LoRa network supports three different categories of devices: end devices, LoRa gateways, and LoRa network servers. LoRa networks use a "star-of-stars" structure. In this architecture, end devices and GWs communicate via LoRa modulation in accordance with LoRaWAN. The GWs transfer raw LoRaWAN frames from the end devices to a network server over a backhaul interface, often over 3G or Ethernet. Thus, gateways can be thought of as two-way relays that execute protocol conversion. The packets that should be sent back to the devices are created by the network server after decoding the packets sent by the devices. LoRaWAN uses an IEEE 64-bit extended unique identifier (EUI) to automatically link IPv6 addresses to LoRa nodes. LoRaWAN uses many layers of encryption to offer security, including (i) a unique network key to secure the network layer, (ii) a unique application key to provide end-to-end security at the application layer, and (iii) a device-specific key to assure secure joining of a node to the network. To take part in a LoRaWAN network, each end device needs to be customized and turned on. When an end device is deployed or reset, it can be activated over the air (OTAA) or by activation by personalization (ABP), which combines the two phases of end-device personalization and activation into a single process. Before exchanging data with a network server during an Over The Air Activation process, end devices must execute a network join protocol. Before the join operation, end devices must be customized with the necessary information, including a globally unique end-device identity (DevEUI), the application identification, and the device name (AppKey).

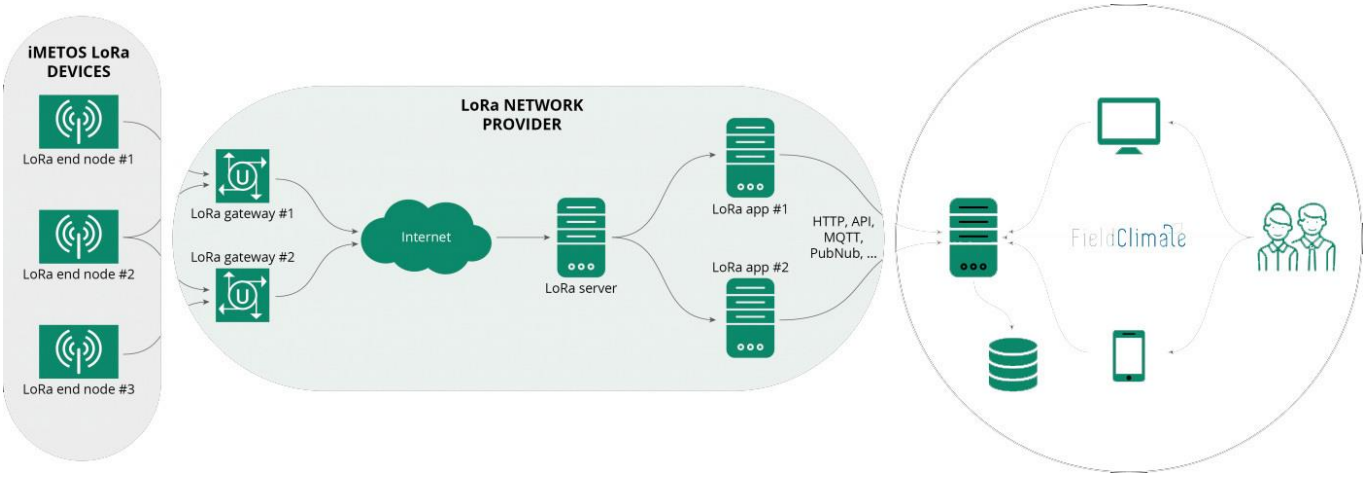

Figure 2.1 LoRaWAN Architecture

The main elements of the project consist of:

• Physical Entity: The thing in the physical world.

• Model: It creates a virtual representation of the physical thing to analyze and forecast the behavior of the model using real-time data from actuators and sensors.

• Analytics: The models are coupled with mathematical models in order to better regulate the behavior of the device.

• Visualization: Pair plot of the datasets parameters.

2.1 LoRaWAN Classes

LoRaWAN supports three different classes of end devices, namely Class-A, Class-B, and Class-C, to meet the needs of varied applications.

2.2 Class-A

All LoRa devices must support the default functional mode of LoRa networks, which is defined by Class A devices [5]. Class A devices support bidirectional traffic, however end devices can schedule an uplink transmission based on their own requirements [4]. In a Class-A network, transmission is always started very asynchronously by the end devices. In order to receive any commands or returned data packets from the network server and to strengthen channel resilience, each uplink transmission is followed by two brief downlink windows. On the other hand, downlink transmission can occur after an uplink broadcast and as a result, a waiting interval is necessary. The least power-hungry devices—Class-A ones—provide less flexibility for downlink broadcasts [1]. The monitoring applications that require a control station to gather data produced by end devices are the primary target applications for Class-A networks.

2.3 Class-B

Bidirectional traffic is supported by Class-B devices with planned receive slots [1]. Class-B devices open additional receive windows at predetermined intervals. The network server is able to determine when the end devices are listening thanks to a synchronized beacon from the gateway. Class-B end devices can receive downlink data or command packets from the network server regardless of the uplink traffic, thus separating uplink and downlink transmission, in contrast to Class-A devices. The principal applications for Class-B devices are those that need receiving commands from a remote controller.

2.4 Class-C

With maximum receive slots, Class-C devices offer bidirectional traffic [1] . While offering nearly constant receive windows, Class-C end devices use the most power of the three device classes. Power grid is one potential use.

2.5 Gateway

The network component known as a gateway (GW) serves as a link between endpoints and network servers. The GWs use an IP backhaul link to forward packets created by end devices

to a network server (e.g: Ethernet or 3G). In a LoRa network, more than one GW can be installed, and different GWs can each receive a single packet. The packets that the network server receives from the end devices are decoded. If the packets need to be transmitted back to the devices, the network server will additionally de-duplicate them.

2.6 MQTT Access to TTN Server

An MQTT interface is available through TheThingsNetwork (TTN) to access the data uploaded using LoRaWAN. We have to publish the data to the broker with specific Topic and the topic name is depend on the user. Topics are hierarchical UTF-8 strings. Each level in a topic is delimited by a forward slash. Every message from a publisher must include a topic. To receive the published message, the entity that consumes the message must subscribe to the same topic. A broker sends the message it receives only to those clients that have subscribed to the same topic. To publish the data in TTN server we have to know server host public address, port number, username and password. We get all the information in the TTN webpage under the end devices application. The querying device must have a MQTT client installed in order to accomplish this. There are MQTT clients available for many different platforms and programming languages. The clients Subscribe the data from the broker and get the data in json format.

2.7 Maximum packet size

The control field is part of the fixed header, while the packet length field is part of the variable header. For messages that are fewer than 127 bytes, a packet length field must have a minimum size of 1 byte. 256 MB is the maximum allowed packet size [2]. With only a single control field and a single packet length field, the minimal packet size is only 2 bytes. A 1-byte packet length field is present in smaller packets of less than 127 bytes. The packets that are between 127 and 16383 utilize two bytes, and so on. Seven bits are used, with the continuation bit being the eighth bit.

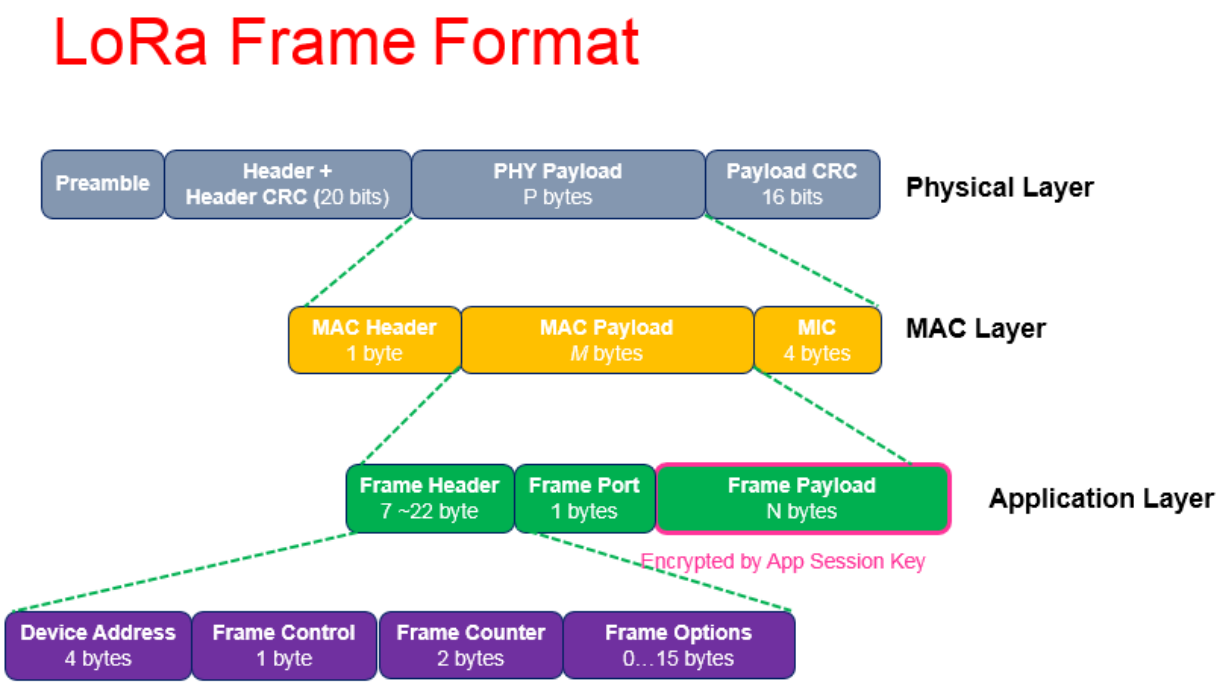

Figure 2.2 LoRa Frame Format

2.8 Frame Structure

Physical Layer Frame: A LoRa frame begins with a preamble at the PHY layer. The preamble describes the packet modulation method in addition to serving as a synchronizing function, and it is modulated using the same spreading factor as the rest of the packet. The preamble typically lasts 12.25 Ts. The rest of the frame is encoded using the code rate provided in the PHY Header after the preamble, which is followed by a PHY Header and a Header CRC that together are 20-bits long and are encoded using the most dependable code rate of. Payload length and whether the Payload 16-bit CRC is present in the frame are two additional details that can be found in the PHY header. In a LoRa network, specifically, only uplink frames contain payload CRC. PHY payload includes a MAC Frame [3].

MAC Layer Frame: A packet's MAC Header, MAC Payload, and Message Integrity Code make up a MAC Layer Frame (MIC). The protocol version and message type, such as whether a frame is a data frame or a management frame, whether it is transmitted in an uplink or downlink, and if it needs to be acknowledged, are defined by the MAC header. This is a vendor-specific message, which the MAC Header can also indicate. The MAC Payload can be swapped out for join request or join accept messages in a join procedure for end node activation. The MIC value is calculated using a network session key and the complete MAC Header and Payload parts. The MIC value is used to authenticate the end node and prevent message forgery [3].

Application Layer Packet: The Frame Header, Frame Port, and Frame Payload make up the MAC Payload that is handled by the Application layer. The type of application determines the Frame Port value. An application session key (App SKey) is used to encrypt the value of the Frame Payload. The AES 128 algorithm is the foundation of this encryption. [3]

3 Evaluation of Event Driven Architecture

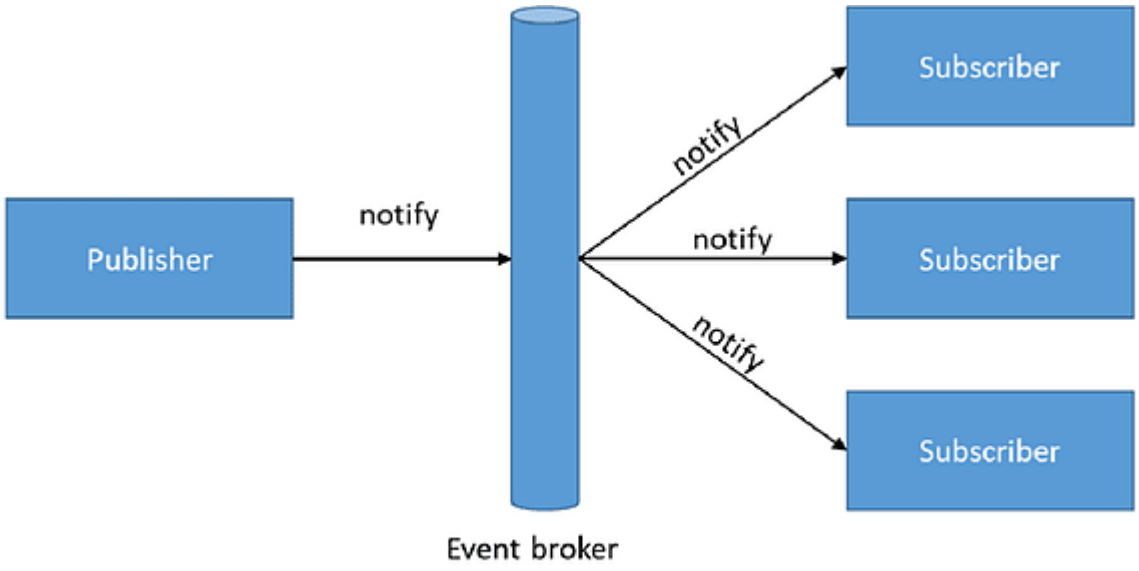

Figure 3.1 Architecture of Event Driven Platform

Event driven architectures provide the foundation for this project's implementation of the streaming platform. Actions are events. An event occurs whenever something occurs, whether it takes place inside or outside of your company. The list of possible activities is endless and includes everything from a customer purchase to an inventory update in a warehouse to an attempted security intrusion.

Companies seek to design their infrastructure with the idea of capturing and responding to time-sensitive or mission-critical events since these occurrences frequently occur. When events cause responses within the company, it is said that the firm is event-driven. IT systems that record, process, and react to events based on predetermined logic handle events in the digital age.

Events in system design have the following traits in common:

• They serve as a record of past events.

• They are unchangeable; they cannot be added to or removed.

• They are persistent, which means that events can be saved and accessed continuously.

• There is no restriction on how often a service can process an event, therefore they can be consumed an infinite number of times.

It's crucial to remember that an event is only an action. Event notifications, which are

messages verifying that an action has been taken, are handled by event-driven architectures. The notification could provide extra information about the event that downstream systems use to apply their business logic, or it could just be a confirmation of the action [4] [5].

For instance, a buy notice may contain the customer's ID, product ID, purchase total, date, and time of the purchase, as well as additional information that a CRM (**Customer Relationship Management)** may utilize to build a customer profile and forecast future product needs.

Event-driven architecture is sometimes referred to as message-driven architecture because it is the event notification or message that is being handled in the system.

A decoupled architecture is used in event-driven architecture (EDA), a system design technique, to record, transmit, and process events. This implies that systems can share information and complete tasks without being aware of one another.

Decoupling, a key element of EDA, provides a significant advantage over alternative data-sharing strategies like APIs. The application must be aware of the dependencies and parameters before making an API call in order to send a correctly structured request. Then it waits for the reply before moving on. This implies that every time data is transferred, there will be a request and a response.

With decoupling, the event router, also known as a broker, decides where to transmit the notice once the event producer sends it without knowing what would happen next. An event router functions similarly to a WiFi router in that it downloads data from the internet and distributes it instantly to various devices. Consumers, often referred to as sinks, automatically get the event to process. Events are processed asynchronously because each service that consumes the event operates independently.

Events can be transported to multiple applications where they can be used for a variety of actions and analyses, which results in event driven architectures having advantages like lower operational costs, greater resilience to failures, and scalability. Additionally, it improves responsiveness because more tasks may be completed concurrently and the most crucial information is handled carefully and quickly. Different components can operate on different workstations and can be independently created in a variety of programming languages. Even if it offers a lot of advantages, it also has certain drawbacks. It becomes increasingly challenging to handle errors in larger and more sophisticated systems since it is hard to keep track of an event's state and which application is in charge of it. The majority of events are non-deterministic, making it more challenging to control the work flow because the exact timing of the occurrences is unknown. The event-timing should be precisely specified if one event must be processed before another [5].

It is simple to perform unit testing on a system with an event-driven architecture, but scenario testing becomes more challenging as systems get larger because it is more difficult to determine which events cause a given result. In addition, when an unknown event is released, it can be challenging to diagnose the issue. Sensors post their data to a topic or message queue. Models follow particular subjects. Topics are also used to handle connections to

visualization services and links between various sub-models.

In EDA architecture, the publisher/subscriber and event streaming patterns are both frequently used. The project makes use of a publish/subscribe mechanism pattern. The event router in a publisher/subscriber architecture, often known as a pub/sub architecture, keeps track of customers' subscriptions to event channels. The router makes sure the subscriber receives an event once it has been published. Events cannot be replayed, therefore a subscriber who joins an event stream after it has already started cannot access events that have already occurred but it is possible to access earlier events in Kafka. After that, the consumer will start receiving fresh events.

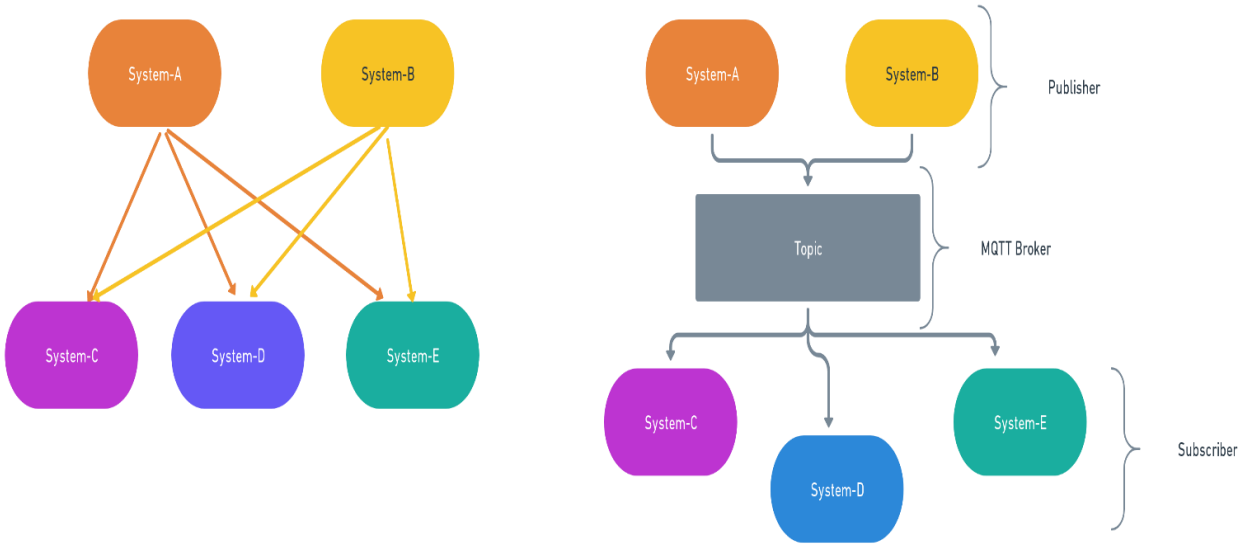

Figure 3.2Comparison of Publish-Subscribe mechanism with Point-to-Point Communication [6]

Let's have a look at the fundamental communication systems of the past to better understand the benefits of the publish-subscribe method. The fundamental software was composed of an input module that transmitted data to the processing module, which then routed the outcome to the output module. As the system expands, there are more input modules and output modules; as a result, an addressing mechanism is utilized to guide the processing module's output to the appropriate output modules. However, this puts extra strain on the system, necessitating the addition of more processing modules to accommodate the load. Due to the necessity for input modules to convey data to the appropriate processing module, this causes a difficulty with routing complexity.

Using the publish-subscribe system, where the identities of the modules are kept secret from one another and they can have well defined duties that they can carry out with ease, this scenario is resolved.

As shown in Figure 3.2, in point-to-point communications between different systems, the

sender must be aware of the address of the receiving computer; as a result, if system A wishes to communicate the same information to systems C, D, and E, it must be sent separately to each individual machine. If the system is small, this type of communication is feasible, but when the sender and receiver grow, the total number of connections grows as well, dramatically increasing the complexity in this type of architecture. As we can see, when System B, a second sender, is also added, the system's total number of connections rises to six. Utilizing a publish-subscribe architecture reduces this complexity by reducing the number of connections to five, as illustrated in Figure 3.2, as well as making communication more easier to manage and grow. There are many benefits to using the publish-subscribe technique. When a system gets bigger, it makes integration and scaling easier. Additionally, it guarantees dependable message delivery even if the architecture of the message broker changes. The ability to fairly easily track topics among publishers and subscribers makes it easier to debug the software module.

This type of architecture would not be effective for systems conducting periodic operations or performing time-sensitive tasks because the publish-subscribe method is an asynchronous messaging system. Additionally, it is not appropriate for media streaming because this type of service requires smooth user-side rendering, which is best accomplished with point-to-point communication methods [6] .

3.1 MQTT as Event-Driven Architecture

A simple open-source messaging network protocol called MQTT is used to send messages between devices. It uses a publish/subscribe communication mechanism to give network clients with limited resources an easy way to distribute telemetry data in low-bandwidth settings.

MIOTY is a low-power, wide-area network (LPWAN) protocol designed specifically for large-scale industrial and commercial IoT deployments. MIOTY protocol is designed to conquer the scalability, interference and mobility issues of legacy wireless IoT technologies [7].

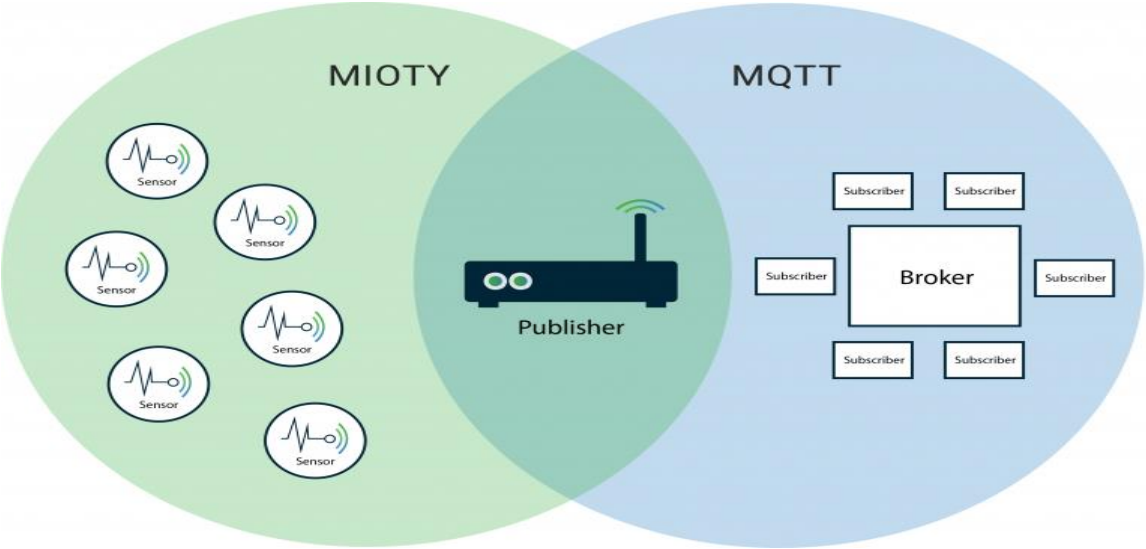

Figure 3.3 MQTT Architecture [7]

3.1.1 Architecture

MQTT is a fairly straightforward publish/subscribe protocol created for quick M2M communications. The MQTT broker and MQTT client are the two essential components of the MQTT architecture.

MQTT broker (server): An MQTT broker or server is computer software that processes messages from publishers, which are external sources, and sends them to subscribers, which are the intended recipients. The computer could be a Raspberry Pi, a desktop PC that is located on-site, or a cloud-based server that runs either open-source or paid software. The Mosquito broker is one of the most well-liked open-source message brokers. On your personal computer, you can run a copy of Mosquito on your own. However, there are widely available cloud-based servers that implement the Mosquito broker, such as Eclipse IoT, ThingMQ, CloudMQTT, and Heroku, rather than running the Mosquito on a local PC. When you want to be able to control your Internet of Things projects remotely, they are quite helpful.

Depending on the implementation, a broker can control thousands of MQTT clients that are connected at once. Consequently, you should take into account aspects like scalability and integration when selecting a MQTT broker. The broker offers additional features in addition to receiving and forwarding messages to clients, such as:

Quality of Service (QoS): The MQTT protocol can offer extra messaging qualities of service through the QoS feature, ensuring that the message is delivered as needed by the service while it is in transit.

Store and Forward: As the name suggests, MQTT enables the broker to store persistent messages.

Security: For connection security, MQTT brokers may request username and password authentication from customers. The TCP connection may be secured using SSL/TLS to ensure the confidentiality of communications while they are in transit.

MQTT client (publishers and subscribers): An MQTT client is any device or program that connects to a MQTT broker across a network, ranging from a basic Arduino microcontroller to a full-featured cloud-hosted application server. It is possible for MQTT clients to be both publishers and subscribers. The same MQTT client can perform both of these tasks. A client can carry out connect, publish, subscribe, unsubscribe, and disconnect activities as necessary during the communication phase [8].

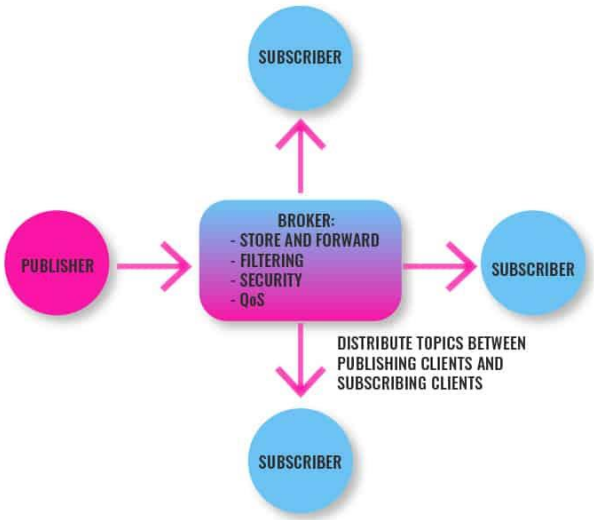

Figure 3.4 MQTT Publisher and Subscriber

Data transfer between clients and the broker is made reliable by MQTT's heavy reliance on the TCP protocol. The connection credentials are transmitted in plain text. Data in transit can be encrypted and secured using SSL/TLS, though. Unencrypted MQTT port defaults to 1883, while encrypted MQTT port defaults to 8883.

MQTT publish-subscribe model works: The MQTT communication model routes data through a centralized server known as the broker to prevent direct connections between devices. Since new devices only need to communicate with the broker, they don't actually need to be compatible with the other clients, which is highly desirable in the IoT because it makes it simple to add new devices without changing the existing infrastructure [8].

A publish occurs when a device (or client) wishes to communicate data to a server (or broker). A subscription is what is used when the process is performed backwards. The broker allows several clients to connect and subscribe to issues that interest them. Through the broker, clients can also post messages on particular subjects of interest. The broker serves as a standard interface for connecting to and exchanging data between devices [8].

Consider a scenario where a humidity sensor needs to communicate its readings to the broker. Additionally, a computer or mobile device needs to receive this humidity value on the other end. The MQTT publish/subscribe data flow procedure includes the following

phases to transmit the readings to the receiving devices:

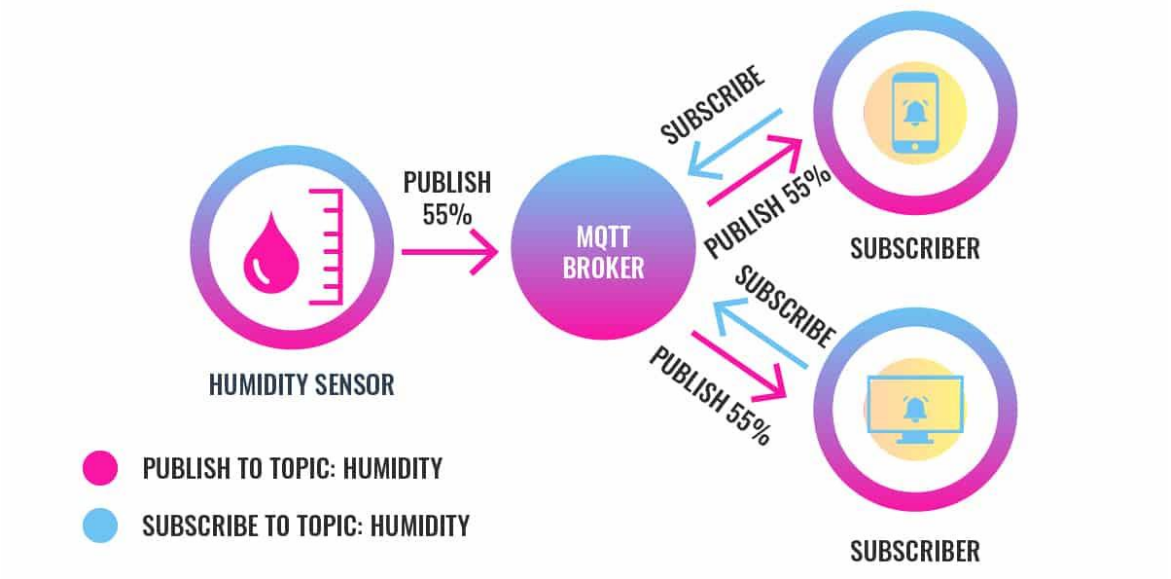

Figure 3.5 Example of Humidity sensor with Publish and Subscriber

The humidity sensor first identifies the subject it wishes to publish on, in this example, "Humidity." After that, it displays the phrase "humidity value."

Second, the computer or mobile device receiving the message subscribes to the topic "Humidity". This enables them to receive the message that the humidity sensor has published–humidity value.

3.1.2 Advantages

• **Packet agnostic:** The payload a packet carries can contain any kind of data. The information could be binary or text. As long as the recipient understands how to interpret it, it doesn't matter.

• **Reliability:** Some Quality of Service (QoS) solutions allow for delivery assurance.

• **Scalability:** The publish/subscribe model scales effectively while using minimal electricity.

• **Decoupled design:** The design features a number of components that isolate the device from the server that is subscribing, resulting in a more effective communication plan.

• **Data consistency**: From source to consumer, data consistency in real-time industrial systems is essential. Decisions may be made incorrectly as a result of outdated or improperly timed data. Data can become erratic in a number of ways. Some messages may be dropped if they arrive at a MQTT broker faster than they can be delivered. Or information from various message streams might be delivered out of order to a client. Additionally, the client might not understand if a value that has not changed is current or stale if a data source goes down. A smart broker can guarantee data consistency by intelligently queuing incoming data and transmitting only the most recent values. Additionally, it can properly order messages

from various data streams by parsing their timestamps, and it can transmit data and connection quality information along with each value update [8].

• **Data security:** When accessing data from a production system, security is crucial. Although many corporate security standards call for isolating OT systems using a DMZ, the MQTT push architecture that links outbound over firewalls is quite secure. Given that messages must flow via two or more servers and that MQTT quality of service guarantees are only valid for a single sender-receiver hop, this presents a challenge for MQTT. Data at the conclusion of a multi-hop daisy chain may thus become suspect. This issue can be resolved with a tunnel and a smart broker that transforms protocols and parses messages. One instance of the smart broker would be connected to the device generating the MQTT data. A secure, TCP-enabled protocol is used to tunnel the message data, along with quality and timestamp data, to a second instance of the smart broker. The values, timestamps, and quality codes would remain intact while the data was converted back into MQTT in that case [8].

3.1.3 Disadvantages

• MQTT employs the TCP protocol, which demands more memory and processing power. TCP employs the handshake protocol, which necessitates regular timeouts for waking up and communicating. This has an impact on battery usage. Additionally, TCP-connected devices often maintain sockets open for one another, which increases memory and power needs.

• The scalability is limited by centralized brokers since each client device incurs some overhead. Scalability is made available by the usage of a local broker hub.

• Because client connections to the broker are always open, a centralized broker could be a point of failure.

• In contrast to HTTP, it is more difficult to implement.

• Advanced features like flow control are not supported.

•Clients for the MQTT protocol must support TCP/IP

3.1.4 Applications

• **Wearables:** To enable low-power Internet of Things wearables, MQTT protocol is frequently employed. Due to its constrained memory and bandwidth capabilities, the MQTT protocol is suited for wearable devices due to its lightweight design. When data is pushed from the wearable device to the server and received by the receiving devices, it is published. MQTT messages may be sent quickly because of the small size of the data being transferred via it, which makes the hardware incredibly responsive.

• **Home automation:** The majority of intelligent home automation systems employ MQTT

as the fundamental communication protocol. For instance, Mark Zuckerberg's custom-built smart home system, named Jarvis (after Jarvis from Iron Man), enables him to manage his house using his iPhone's voice or text capabilities. By connecting to MQTT, the device can respond to orders by sending MQTT messages to a broker. This makes it possible to communicate with any device that supports the MQTT protocol [8].

• **Smart farming:** By providing ongoing monitoring and reporting of weather and soil factors such as soil pH, soil temperature and moisture, air temperature, humidity, and sunlight availability, smart farming has the potential to boost total production and improve the efficiency of farming systems. Even with a problematic Internet connection, the MQTT protocol and IoT sensors for agricultural provide for seamless data gathering and access to cloud servers (broker).

• **Smart Metering**: Compared to conventional meters, smart meters provide more information on energy consumption in the energy and utility sectors. The software used to deploy the meter can include an embedded MQTT client, which can be used to broadcast (publish) data with ensured message delivery to deliver precise meter readings in real-time. MQTT technology can be used by homeowners to limit how much energy is used by domestic equipment. This improves energy management and improves the accuracy of billing.

• **Faraway sensing:** Low-power sensors that operate in areas with shoddy internet connections are frequently utilized to keep an eye on remote situations. Due to its capacity to accommodate IoT sensors with lower-priority data transfer requirements, MQTT is an appropriate protocol for corresponding with such gadgets.

4 System design and installation

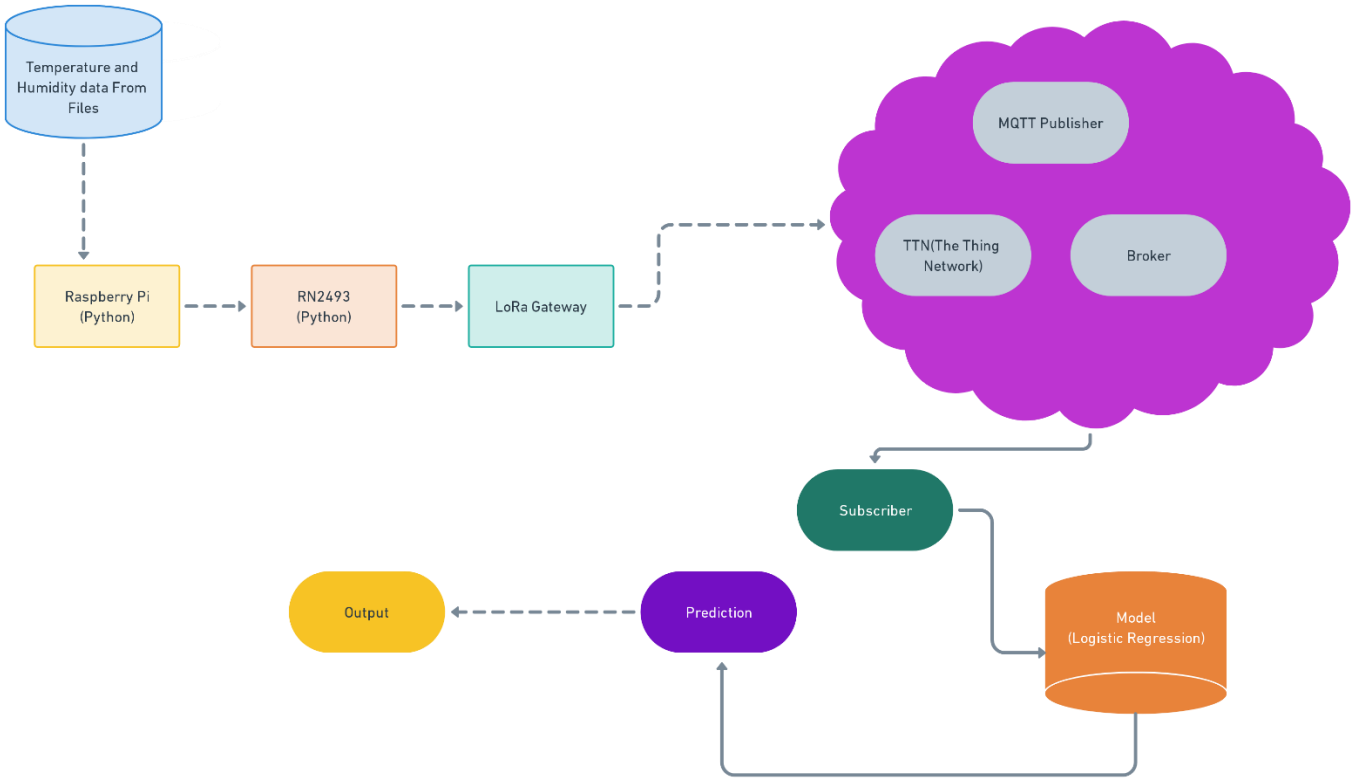

Figure 4.1 Software Architecture

The fundamental idea behind the project is to use a Raspberry Pi to extract data from a sensor file instead of using real or live sensor data.. Using a Python application, we connect the RN2483 sensor to the Raspberry Pi and obtain the data serially from the stored files. We use a LoRa Gateway to transfer the data once it has been obtained. We use Gateway to send 10 temperature readings as a packet in a block. Standard Internet of Things (IoT) protocols can use MQTT to access data. It is quite easy to add several devices to this solution because they can all connect through the Gateway and communicate real-time data using the MQTT protocol. MQTT offers fast throughput and connects to a wide range of applications. Data analysis requires storage of the information in order to analyze it. We transmit the data to the model and forecast the results after collecting all the data.

4.1 Components and Data Flow

4.1.1 Data Source

RN2483 433/868MHz Microchip is an integrated modem with a range of more than 15 kilometers is called the LoRaTM Integrated Module (suburban). An extended battery life of more than 10 years is conceivable due to the low power usage. Millions of wireless sensor nodes can also be linked to LoRa gateways and cloud servers via the Internet of Things. Due to LoRa's distinctive spread-spectrum-based modulation, which enables demodulation of signals 20 dB below the noise level, the system is robust. This improved network efficiency, allows for great sensitivity over very long distances, and eliminates interference. Over 433 and 868MHz are covered by the RN2483 modem. These are the Industry, Scientific, and Medical (ISM) frequency bands, which do not require a license. It performs the function of an end device in the LoRa network architecture.

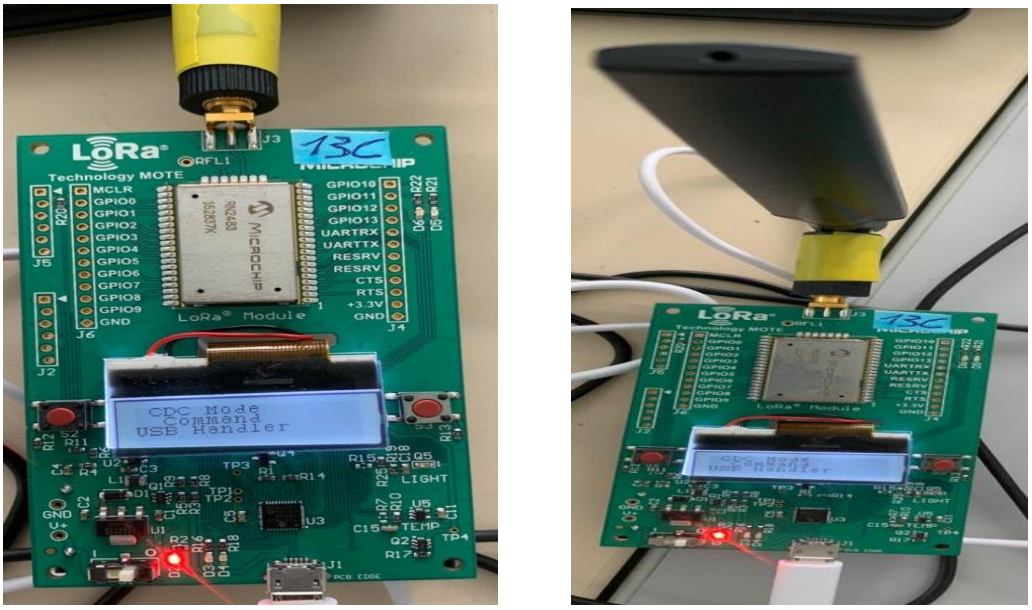

Figure 4.2 RN2483 Sensors

The inbuilt LoRaWAN Class A protocol in the RN2483 Module enables smooth access to any LoRaWAN compliant network infrastructure, whether it is being used publicly or privately. The module's ease of use reduces the amount of time needed for development and accelerates time to market. Low power applications including the Internet of Things (IoT), Machine to Machine (M2M), smart cities, sensor networks, industrial automation, and others are perfect for LoRa technology. The Raspberry Pi 3 was used to read the RN2483 temperature over serial port and then connect it to Ethernet.

4.1.2 Raspberry Pi and Display

Popular and reasonably priced minicomputers include the Raspberry Pi. Through its GPIO ports, USB compatibility, and built-in WLAN support, it is perfectly suited for machine-to-machine solution prototyping. Raspbian, the Raspberry Pi's standard Linux distribution, serves as its operating system.

Figure 4.3 Raspberry Pi and Display

Raspberry Pi 3 Model B+ is 64-bit quad core processor running at 1.4GHz, dual-band 2.4GHzand 5GHz wireless LAN, Bluetooth 4.2/BLE, faster Ethernet, and PoE capability via a separate PoE HAT. The dual-band wireless LAN comes with modular compliance certification, allowing the board to be designed into end products with significantly reduced wireless LAN compliance testing, improving both cost and time to market. The mechanical footprint of the Raspberry Pi 3 Model B+ is identical to that of the Raspberry Pi 2 Model B and the Raspberry Pi 3 Model B [9].

4.1.3 LoRaWAN Gateway

Data transmitted through LoRaWAN radio is received by the LoRaWAN gateway, which digitizes it. The digital data from the sensor is enhanced with meta-data before being transmitted over a TCP/IP connection to the LoRaWAN network server. The sensors receive and transfer data from the LoRaWAN network server in a similar manner. Ethernet, WLAN, or cellular connections are all acceptable TCP/IP connections. Backhaul refers to this link to the LoRaWAN network server.

Due to a lack of key information, the LoRaWAN gateway is unable to decrypt the data. It is unable to ascertain whether the LoRaWAN device is recognized by the network

server it is communicating to and does not get any feedback to this effect. This was done on purpose to prevent anyone from viewing the communication between the server and sensor, not even someone who has physical access to the gateway and might even be able to log in.

Figure 4.4 LoRa Gateway

Based on a Raspberry Pi 3B+, an LRWCCx-MPCIE Concentrator Card, and an SBCPoE RPi adaptor hat, the LRWGW RPi is a LoRaWAN gateway (hence referred to as Gateway).

4.1.4 MQTT Broker

As was already mentioned in section 3.1, MQTT processes streams using an event-driven architecture. The MQTT Publish/Subscribe protocol's central component, the MQTT broker, is a server that receives all messages from MQTT clients and then forwards them to the proper subscribing clients. A key piece of software in the MQTT architecture is a MQTT broker. It functions just like a real estate broker, who first investigates the parties involved and then starts a deal after ensuring that all applicable laws are followed.

The only difference is that a MQTT broker handles message transactions rather than financial ones. Here's how MQTT brokers specifically help MQTT clients do transactions Allow devices (a.k.a. “client devices” or simply “clients”) to make a connection request

• Based on the connection details supplied by the connecting device, authenticate the devices (s)

• Once the device has been authorized, confirm that it can transmit and receive messages securely using Transport Layer Security (TLS) encryption (as one option)

• Saves messages on the server so they can be sent again in the event of an unintended loss of connection, upon client connection, upon client disconnect, etc.

Due to the decoupled communication that a MQTT broker enables, the entire architecture may be grown extremely quickly without even affecting current client devices.

4.1.5 Models

Using probabilistic predictions, the supervised machine learning technique known as logistic regression completes binary classification problems. The model produces a binary or dichotomous result with only two options: true or false, yes or no, or 0/1. Data are categorized into discrete classes using logical regression, which examines the relationship between one or more independent variables. It is frequently used in predictive modeling, where the model calculates the mathematical likelihood that a particular incident falls under a given category.

The model was written in Python Programming language using Machine Learning Library. The model gives the output. In this project

1 = cooling speed is in the expected range

0 = temperature curve deviates from the expected pattern

4.2 Installation overview

On the virtual server that the University has made available, the software components for the MQTT broker.

4.2.1 Installation of MQTT broker

On the virtual server made available by the university.

Installation Steps for client

Pip install paho-mqtt.

5 Curve Fitting

Introduction: Curve fitting's goal is to identify the parameters that go along with a model (function or equation) that theoretically describes experimental data. Mechanistic models are the ones that are most important to us. Mechanistic models are created expressly to shed light on the physical, biological, or chemical process considered to control the event being studied. Mechanistic model parameters are quantitative estimations of actual system characteristics (rate constants, dissociation constants, catalytic velocities etc.). Mechanistic models must be distinguished from empirical models, which are mathematical functions created to match a certain curve but whose parameters are not always related to biological, chemical, or physical properties. Curve fitting has a different goal; one just tries to draw a curve through the baseline. The other is about empirical fitting, which includes smoothing and interpolations like splines.

Finding the best collection of parameters for a specified function that best fits a given set of observations is a sort of optimization known as curve fitting. Contrary to supervised learning, curve fitting calls for the definition of a function that links instances of inputs to examples of outputs. A straight line (linear regression), a curved line (polynomial regression), and many other shapes are possible for the mapping function, also known as the basis function. When an optimization technique is utilized to identify the precise optimal function parameters, this gives the freedom and control to specify the shape of the curve.

Curve fitting is best understood in two dimensions, such as a graph. Consider that we have samples of inputs and outputs for data from the problem area. The independent variable or function's input is represented by the x-axis. The dependent variable or function's output is represented by the y-axis. Although we are unsure of the exact shape of the function that converts examples of inputs into outputs, we believe that we can estimate it using a common function form. It is necessary to define the mapping function's functional form (also known as the basis function or objective function) before looking for the parameters of the function that produce the least amount of error. By utilizing the domain's observations as inputs, sending those inputs to our proposed mapping function, computing the output, and comparing that output to the observed output, error is obtained. After fitting, we can interpolate or extrapolate new points in the domain using the mapping function. It is typical to use the mapping function to calculate a series of outputs from a set of input values, then display the results on a line to illustrate how the output varies with the input and how well the line fits the observed points. The shape of the mapping function is crucial for curve fitting [10].

5.1 Categories

Based on how we treat the observed data, curve fitting typically falls into one of two groups [11].

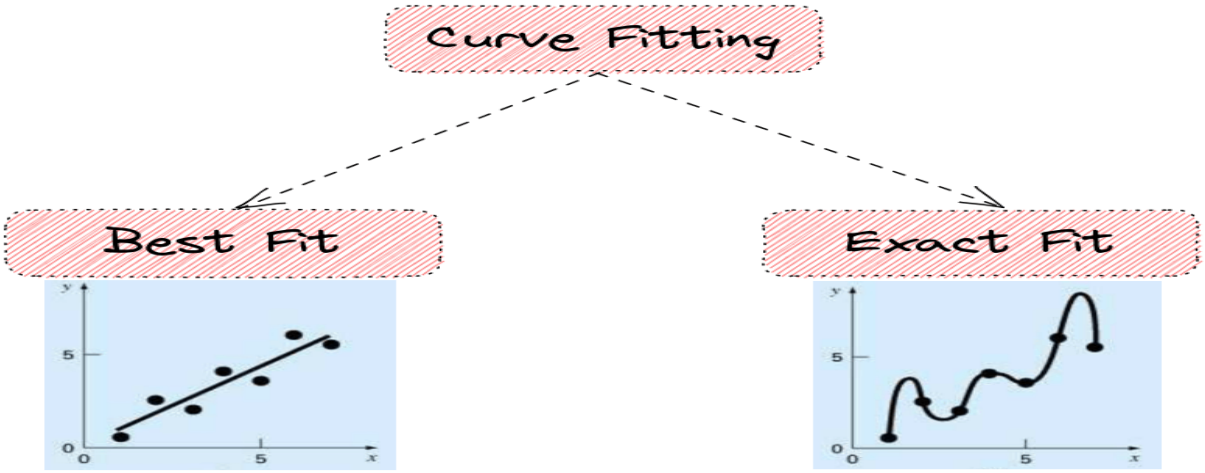

Figure 5.1 Category of Curve Fitting

Best Fit: In this situation, we presumptively use noisy measured data points. So, fitting a curve that intercepts each data point is not what we want to achieve. Our objective is to discover a function that, when applied to the provided data points, minimizes a preset error.

Linear regression, where the curve is a straight line, is the best fit method that is the simplest. More specifically, we have a collection of samples (x 1, y 1), (x 2, y 2), (x n, y n) and the parametric function y = an x + b, where an is the slope and b is the intercept. In order to reduce an error threshold on the provided samples, our objective is to learn the values of a and b.

Exact Fit: In this scenario, we assume that the provided samples are not noisy and aim to discover a curve that traverses each point. It is helpful when trying to determine a function's minimum, maximum, or zero crossings or to generate finite-difference approximations.

We may see an illustration of a precise fit using polynomial interpolation in the figure below.

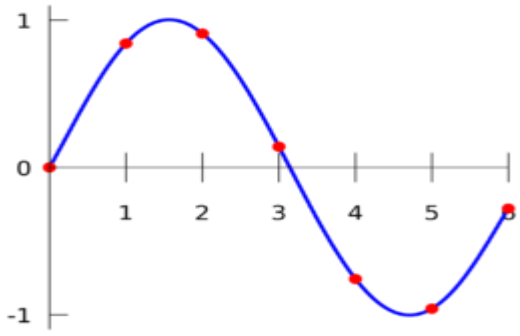

Figure 5.2 Exact Fit Curve Fitting

5.2 Least-Squares Algorithm

For curve fitting, numerous algorithms have been proposed. The most popular approach is known as least squares, where we look for a curve where the sum of the squares of the residuals is the smallest. We refer to the discrepancy between the observed sample and the estimation from the fitted curve as the residual. Both linear and non-linear relationships can use it.

We wish to fit the function y = a * x + b. Where y is the calculated output, x is the input, and a and b are parameters of the mapping function found using an optimization algorithm.

T = $\sum_i y_i$ - (b + a x_i))^2 by determining the parameters a and b. (sum of residuals).

There are two requirements to decrease the aforementioned term:

$$\frac{\partial T}{\partial a} = 0 \rightarrow 2\sum_i (y_i - (b + ax_i))(-x_i) = 0 \rightarrow \sum_i x_i y_i = b\sum_i x_i + a\sum_i x_i^2$$
$$\frac{\partial T}{\partial b} = 0 \rightarrow 2\sum_i (y_i - (b + ax_i)) = 0 \rightarrow \sum_i y_i = nb + a\sum_i x_i$$

The Gauss-Newton method is one of the most well-known algorithms for non-linear least-squares problems. Although we lack a theoretical demonstration that the method converges, it does so in practice and is simple to use.

5.3 Exponential Curve Fitting

Different types of exponential growth and/or decay curves exist. Most crucially, depending on what influences their decay/growth tendency, things can degrade or grow mono- or multi-exponentially [12].

Consider the case when we have a general exponential function of the following kind and are confident that it accurately describes our data (where a, b, and k are fitting constants):

y = a$* e^{kx} + b$

Here, “y” is the output, “x” is the input, “a” is Pre-exponential factor, “k” is the rate constant and “b” is Bias.

In order for curve fit to use the exponential function while performing the fitting, we must first define it as indicated above.

The scipy.optimize.curve fit() function in Python Scipy is used to determine the parameters that provide the best fit for a least-squares fit. Our model is adjusted by the curve fit method to the data.

Finding the ideal set of parameters for the specified function that best fits the given set of observations requires the use of the curve fit.

5.4 Curve Fitting Python API

Python can be used to fit curves to our dataset. The curve fit() function of the SciPy open source library allows curve fitting using nonlinear least squares. The identical input and output data are passed as arguments to the function, along with the name of the mapping function to be used. The mapping function must accept a certain number of arguments as well as instances of input data. The coefficients or weight constants for the remaining arguments will be determined by a nonlinear least squares optimization procedure.

Exponential Fitting:

Consider the case when we have a general exponential function of the following kind and are confident that it accurately describes our data (where a, b, and c are fitting constants):

```
def exponential(x, a, k, b):
   return a*np.exp(x*k) + b
```

We must input this function into a scipy function same like before.

Fit the exponential data

constants = curve_fit(exponential, x_array, y_array_exp, p0=[40,-0.05, 13])

Inputs:

f — function used for fitting (in this case exponential)

x_array — array of x-data for fitting

y_array_exp — array of y-data for fitting

p0 — array of initial guesses for the fitting parameters.

Outputs:

constants — array of parameters from fit.

a = constants[0][0]

k = constants[0][1]

b= constants[0][2]

a : Pre-exponential factor

k : Rate constant

b: Bias

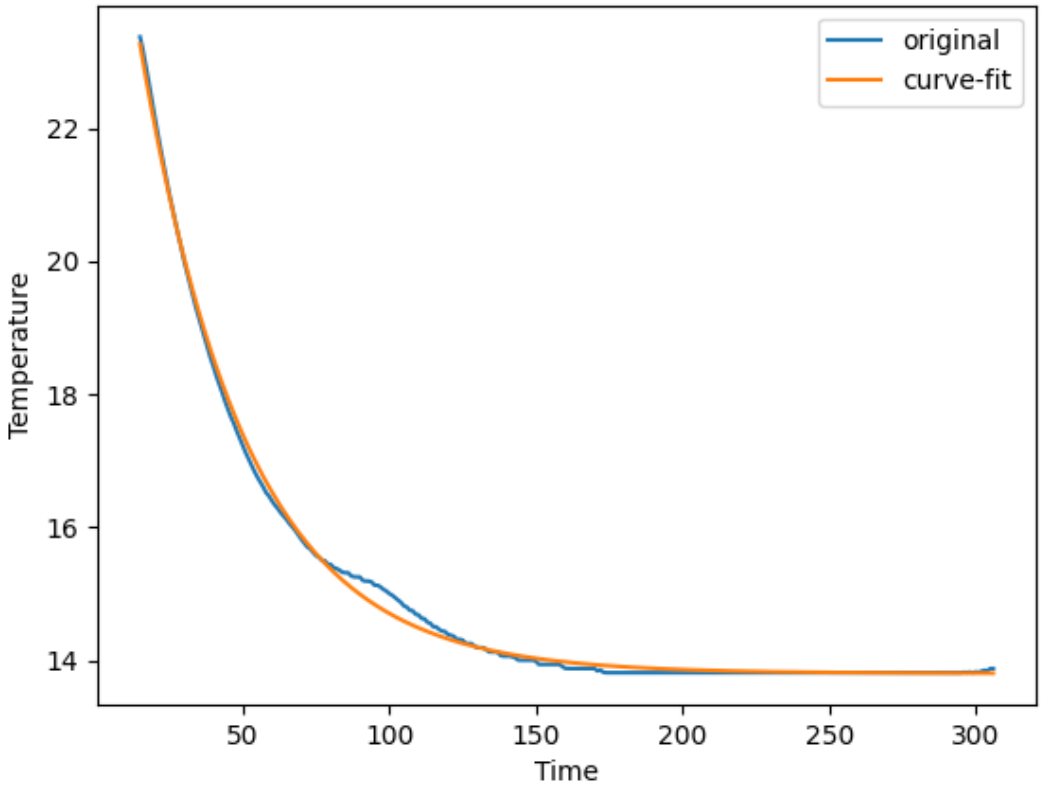

Figure 5.3 Original data vs Curve fitting data

From the outputs of the curve fit function, we can extract the parameters and their standard deviations. We can then compute the residuals by deducting the calculated value (from our fit) from the actual observed values.

Mean Squared Error (MSE): The amount of error in statistical models is measured by mean squared error (MSE). The average squared difference between the observed and predicted values is calculated. When there is no error in a model, the MSE is zero. As model error increases, so does its value. The mean squared deviation is another name for the mean squared error (MSD).

The formula for MSE is the following.

$$MSE = \frac{\sum(y_i - \hat{y}_i)^2}{n}$$

Where:

y_i is the i^{th} observed value.

$\hat{y}_i$ is the corresponding predicted value.

n = the number of observations.

Mean or Average: In math and statistics, the mean is a single number that represents the data's center point or typical value. It is the most common measure of central tendency and is also known as the arithmetic mean. It is frequently referred to as the "average."

The formula for Mean is the following.

$$\bar{x} = \frac{\sum x_n}{n}$$

Where:

$\bar{x}$ is the sample average of variable x.

$\sum x_n$= sum of n values.

n = number of values in the sample.

Label: In my project we had 47 data files. We had 12 bad cooling data files among those, with the remaining data files being good cooling. I just gave the label bad for bad cooling data and good for good cooling file.

The parameters table for the datasets are given below.

file	a_para	k_para	b_para	mse	avg	label
Ex_10_Box_101.json	58,2799	-0,0752	13,2672	0,0029	13,4393	good
Ex_10_Box_102.json	40,6524	-0,0434	13,7481	0,0044	14,4134	good
Ex_10_Box_146.json	52,4257	-0,0632	13,4046	0,0034	13,6909	good
Ex_10_Box_154.json	34,9609	-0,0407	13,6479	0,0037	14,3211	good
Ex_10_Box_198.json	29,5467	-0,0318	13,9032	0,0057	14,9108	good
Ex_10_Box_199.json	28,2679	-0,0295	14,0172	0,0081	15,1453	good
Ex_10_Box_201.json	57,571	-0,0659	13,3605	0,0029	13,6328	good
Ex_10_Box_203.json	26,0303	-0,0305	14,134	0,0031	15,1035	good

Ex_10_Box_204.json	25,0409	-0,0316	13,9684	0,0011	14,834	good
Ex_10_Box_206.json	25,3258	-0,0214	14,3472	0,0168	16,2169	bad
Ex_10_Box_214.json	24,4891	-0,0182	14,4425	0,0419	16,8265	bad
Ex_10_Box_219.json	25,5534	-0,0218	14,2797	0,0209	16,1003	bad
Ex_10_Box_59.json	36,6897	-0,0403	13,7238	0,005	14,4463	good
Ex_10_Box_60.json	36,661	-0,0439	13,6691	0,0063	14,2509	good
Ex_10_Box_61.json	16,9883	-0,0217	14,0272	0,0065	15,2515	bad
Ex_7_Box_132.json	11,1309	-0,0323	14,6005	0,142	15,2844	good
Ex_7_Box_133.json	12,3004	-0,023	14,9422	0,103	16,1336	good
Ex_7_Box_141.json	12,3886	-0,0193	14,6864	0,0832	16,1811	bad
Ex_7_Box_142.json	13,0877	-0,0193	14,4546	0,0603	16,0378	bad
Ex_7_Box_150.json	12,8746	-0,0277	14,5604	0,0995	15,5368	good
Ex_7_Box_151.json	13,5765	-0,0236	14,7889	0,0891	16,0641	good
Ex_7_Box_154.json	12,8997	-0,0242	14,7361	0,1077	15,9054	good
Ex_7_Box_156.json	52,6143	-0,0001	-37,6367	0,0736	14,1167	good
Ex_7_Box_159.json	13,3112	-0,0379	14,4806	0,1403	15,1307	good
Ex_7_Box_160.json	13,2292	-0,0359	14,5041	0,1177	15,2053	good
Ex_7_Box_161.json	3,4211	-0,0021	11,7306	0,0826	14,1154	good
Ex_7_Box_165.json	12,5281	-0,0427	14,2702	0,1345	14,7814	good
Ex_7_Box_168.json	12,5045	-0,0182	14,5997	0,0686	16,2293	bad
Ex_7_Box_170.json	13,8495	-0,0143	14,9735	0,0515	17,3623	bad
Ex_7_Box_171.json	12,8977	-0,0163	14,8973	0,0608	16,8078	bad
Ex_7_Box_172.json	12,8176	-0,0284	14,7922	0,1001	15,7352	good
Ex_7_Box_173.json	12,8406	-0,0408	14,4871	0,1405	15,0503	good
Ex_7_Box_174.json	12,2328	-0,0264	14,8759	0,1499	15,8696	good
Ex_7_Box_175.json	9,7332	-0,0108	14,7159	0,1754	16,9968	bad
Ex_7_Box_176.json	9,4216	-0,0063	13,8516	0,1691	17,4628	bad
Ex_7_Box_177.json	12,9205	-0,0179	14,6959	0,0522	16,4095	bad
Ex_9_Box_132.json	15,4108	-0,0263	13,9599	0,0053	15,3317	good
Ex_9_Box_133.json	15,2291	-0,025	13,9459	0,0044	15,3937	good
Ex_9_Box_135.json	12,1781	-0,0372	13,5955	0,0315	14,2498	good
Ex_9_Box_136.json	14,2565	-0,031	13,7687	0,0159	14,7727	good
Ex_9_Box_137.json	14,7859	-0,0321	13,793	0,0163	14,7855	good
Ex_9_Box_146.json	13,4244	-0,0245	13,8771	0,0054	15,1892	good
Ex_9_Box_147.json	14,3299	-0,0277	13,8045	0,0106	14,9926	good
Ex_9_Box_148.json	11,1433	-0,0369	13,5192	0,0382	14,1243	good
Ex_9_Box_149.json	14,1183	-0,026	13,8403	0,0073	15,1149	good
Ex_9_Box_153.json	14,478	-0,0229	13,9048	0,0062	15,4575	good
Ex_9_Box_155.json	14,5434	-0,0213	14,0801	0,0056	15,7937	good

Table 1 Parameters of the Dataset

Pair plots: Pair plots are used to determine the most distinct clusters or the best combination of features to describe a connection between two variables. By creating some straightforward linear separations or basic lines in our data set, it also helps to create some straightforward classification models.

There are several other graphs that you can use with the Exploratory Data Analysis (EDA) technique to help you understand the relationship between the various factors in the data better.

All of the aforementioned methods will assist you in selecting the optimum variables for your machine learning model. Here is the pair plot between the parameters which are shown in the pictures.

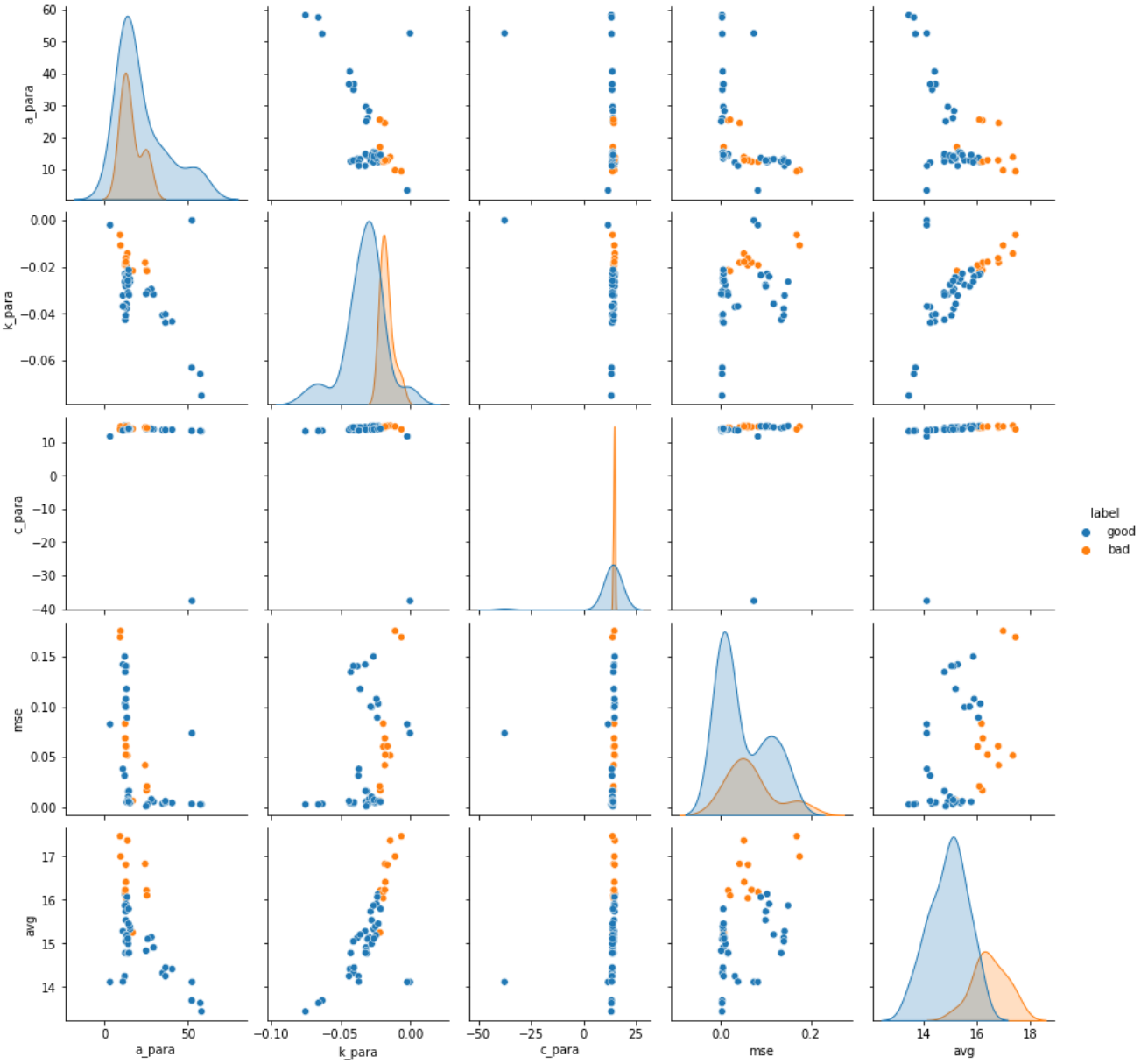

Figure 5.4 Pair Plot of the Parameters

We can clearly distinguish between good and bad labels for the "k" parameters (rate constant), "c" parameters (bias), and average parameters from the preceding graph. Accordingly, those parameters are more crucial than the "a" parameter (Pre-Exponential Factor) and the mse (Mean Square Error) parameters.

Correlation matrix: Correlation describes the relationship between two or more variables. These variables could be input data features used to predict/classify our target variable. Correlation is a statistical technique that determines how one variable moves/changes in relation to another. It gives us an idea of the strength of the relationship between the two variables. It is a bivariate analysis measure that describes the relationship between various variables. The correlation matrix or correlation table is an analysis tool that brings together correlation coefficients between an x-axis and a y-axis where we find different variables.

There are certain characteristics of the correlation given below

- We can anticipate one variable from the other if two variables are highly corelated
- Correlation is critical in identifying the important variables on which other variables rely.
- An accurate correlation analysis improves data comprehension.

The correlation matrix can produce one of three outcomes:

A positive correlation: There is a relationship between these two variables since they move in the same direction.

A neutral correlation: There is no connection between the two variables in a neutral correlation.

A negative correlation: The two variables move in the opposite direction when there is a negative connection.

The correlation between the parameters is given below:

Out[44]:

	a_para	k_para	c_para	mse	avg	label
a_para	1.000000	-0.596995	-0.381262	-0.481441	-0.580313	0.210189
k_para	-0.596995	1.000000	-0.272413	0.308045	0.673924	-0.478696
c_para	-0.381262	-0.272413	1.000000	-0.020601	0.240393	-0.115327
mse	-0.481441	0.308045	-0.020601	1.000000	0.435287	-0.158150
avg	-0.580313	0.673924	0.240393	0.435287	1.000000	-0.714570
label	0.210189	-0.478696	-0.115327	-0.158150	-0.714570	1.000000

Figure 5.5 Correlation between the Parameters

The correlation matrix helps to predict the evolution of the relationship between the variables. Linear regression and Logistic regression will perform poorly if we have highly correlated attributes that means, low correlation is better for the machine learning algorithm. According to this correlation matrix, the "a" parameters has a significant negative

correlation with the "k" parameters and a weak negative correlation with the bias parameters.

5.5 Pros and Cons of Curve Fitting

Advantages: If you are familiar with the mathematical model of the process that generated your data, you should be able to estimate its parameters quite accurately and get insight into it. A parametric regression was used.

Disadvantages: In order to obtain an acceptable fit, you must first have a model of your system that is reasonably representative and an initial estimate of your parameter set that is reasonably accurate (at least with respect to orders-of-magnitude).

6 Machine Learning Model

We are using different machine learning models for my project. We discussed different machine learning algorithms below.

6.1 Logistic regression

Another potent supervised machine learning approach utilized for binary classification issues is logistic regression (when target is categorical). The best approach to conceptualize logistic regression is as a linear regression applied to classification issues. In essence, logistic regression models a binary output variable using the logistic function given below. The main distinction between logistic regression and linear regression is that the range of logistic regression is constrained to values between 0 and 1. Additionally, logistic regression does not require a linear relationship between the input and output variables, in contrast to linear regression. This is because the odds ratio underwent a nonlinear log change [13].

Logistic Function = $\frac{1}{1+e^{-x}}$

The input variable in the logistic function equation is x. Let's supply the logistic function with values ranging from -20 to 20. The inputs have been changed to values between 0 and 1, as seen in Figure.

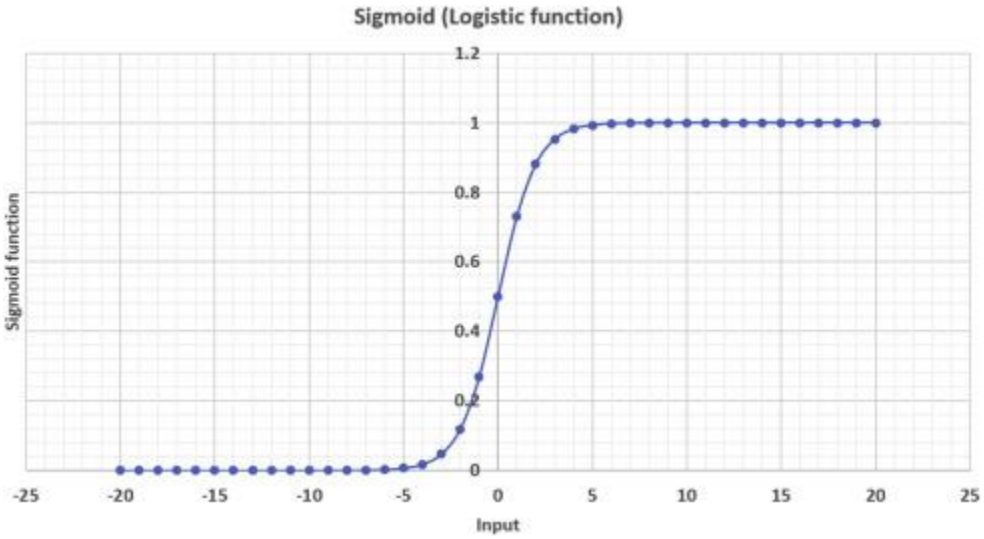

Figure 6.1 Logistic regression applied to a range of −20 to 20

Maximum Likelihood Estimation (MLE), which is a conditional probability, is utilized as the loss function in logistic regression as opposed to linear regression, which employs MSE or RMSE as the loss function. Predictions will be labeled as class 1 if the probability is greater than 0.5. If not, class 0 will be chosen.

6.1.1 Types of logistic regression

There are three different kinds of logistic regression:

Binary logistic regression: When the dependent variable (Y) is binary in nature, binary logistic regression is the statistical method used to forecast the relationship between the dependent variable (Y) and the independent variable (X). The result could be Success/Failure, 0/1, True/False, or Yes/No, for instance. In this project, we have been concentrating on this particular kind of logistic regression.

Multinomial logistic regression: When there is a categorical dependent variable with two or more unordered levels, multinomial logistic regression is utilized (i.e two or more discrete outcomes).

Ordinal logistic regression: Ordinal logistic regression is used when the dependent variable (Y) is ordered (i.e., ordinal). The dependent variable has a meaningful order and more than two categories or levels.

6.1.2 Advantages and Disadvantages of using logistic regression

Advantages of logistic regression:

- Particularly in the context of machine learning, logistic regression is substantially simpler to implement than other approaches.
- When the dataset can be linearly separated, logistic regression is effective.
- In addition to providing a measure of the importance of an independent variable (i.e., the coefficient size), logistic regression also reveals the direction of the association (positive or negative).

Disadvantages of logistic regression:

- A continuous result cannot be predicted by logistic regression.
- In a logistic regression, the relationship between the predictor (independent) variables and the dependent (predicted) variable is assumed to be linear. In the real world, it is highly unlikely that the observations are linearly separable.
- If the sample size is too small, the results of logistic regression could be inaccurate.

6.2 Logistic regression using scikit-learn

The procedures for computing a logistic regression line with Sklearn

First, import "train_test_split," from the sklearn library and define a seed number of 43, and use a 75/25 split randomly (75% is used for training and 25% will be used for testing randomly).

```
from sklearn.model_selection import train_test_split
```

np.random.seed(43
X_train, X_test, y_train, y_test=train_test_split(x_features,y, test_size=0.25)

This class uses the 'liblinear' library, the 'newton-cg','sag','saga', and 'lbfgs' solvers to implement regularized logistic regression.

newton-cg: A Hessian matrix is used by the Newton method known as newton-cg. A square matrix of second-order partial derivatives of a scalar-valued function is used in the Hessian matrix, which was created in the 19th century by the German mathematician Ludwig Otto Hesse.
Sag: "sag" is stochastic average gradient descent.
Saga: "saga" is an extension of "sag" that allows for L1 regularization.
Lbfgs: lbfgs stands for limited-memory Broyden–Fletcher–Goldfarb–Shanno. lbfgs is an iterative method for solving unconstrained nonlinear optimization problems which approximates the second derivative matrix updates with gradient evaluations.

Note: Keep in mind that regularization is used by default.

let's import the logistic regression library and call it "lr.

from sklearn.linear_model import LogisticRegression
lr=LogisticRegression()

Let's apply the lr model to fit (X_train,y_train) as follows:

lr.fit(X_train,y_train)

Next, let's apply the "lr" trained model to predict the "X_test" and call the resultant array as the "y_pred" as shown below:

y_pred=lr.predict(X_test)

6.3 Support Vector Machine

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. The goal of SVM is to maximize the margin between support vectors using a separating hyperplane, as contrast to many ML algorithms where the goal is to minimize a cost function. However, primarily, it is used for Classification problems in Machine Learning [13].
The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. In Fig. 6.1, a separating hyperplane (solid black line) is drawn to separate blue instances from the red ones. This hyperplane is drawn in a manner to maximize the margin from both sides.

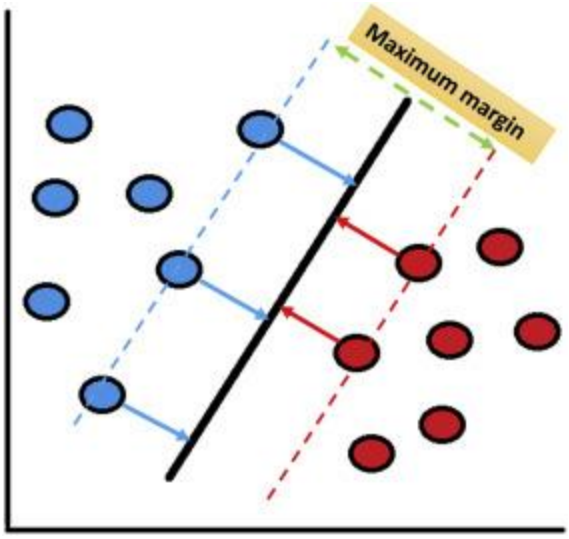

Figure 6.2 Support vector machine for classification illustration

Support vectors are what give this technique its name—they are the blue and red examples that are closest to the separating hyperplane. The "margin" is the separation between the dashed blue and red lines and the separating hyperplane (solid black line). As was indicated earlier, the goal is to increase the margin on both sides. This example is for a two-dimensional space, and for an n-dimensional space, the hyperplane will be n-1.

6.3.1 Types of SVM

There are two types of Support Vector machine

Linear SVM: Linear SVM is used for linearly separable data, which is defined as data that can be divided into two classes using a single straight line. Linear SVM classifiers are used for such data

Non-linear SVM: Non-linear SVM is used for non-linearly separated data, which means that if a dataset cannot be classified using a straight line, it is non-linear data, and the classifier employed is referred to as a Non-linear SVM classifier.

Most of the Real-World applications are non-linear therefore, if the data cannot be separated linearly, the next two ideas are crucial to take into account.

Soft margin: Soft margins are employed to draw a line between classes, although they will overlook the occasional incorrect classification. The parameter "C" in the SVM algorithm denotes this level of tolerance. In essence, the parameter C represents a trade-off between having a large margin and successfully classifying training data. When employing SVM, C acts as a regularization parameter. C is one of the variables that can be utilized to improve grid searches.

Larger C= Smaller width of the margin (Higher training accuracy and lower testing accuracy)

Lower C= Larger width of the margin (Lower training accuracy and higher testing accuracy)

Kernel tricks: These techniques make use of already-existing features and apply various transformation functions to produce new features when the data cannot be linearly separated.

When working with nonlinearly separable data, for instance, linear, polynomial, or radial basis function (rbf) transformations might be utilized.

There are other Kernel functions available in the scikit-learn library, including "linear," "poly," "rbf," "sigmoid," etc. Poly and rbf are the most widely used and often offer the highest accuracy.

6.3.2 Hyperplane and Support Vectors in the SVM algorithm

Hyperplane: In n-dimensional space, there may be several lines or decision boundaries that can be used to divide classes; nevertheless, the optimum decision boundary for classifying the data points must be identified. The hyperplane of SVM is a name for this optimal boundary.

The dataset's features determine the hyperplane's dimensions, therefore if there are just two features (as in the example image), the hyperplane will be a straight line. Additionally, if there are three features, the hyperplane will only have two dimensions.
We always build a hyperplane with a maximum margin, or the greatest possible separation between the data points.

Support Vectors: Support vectors are the data points or vectors that are closest to the hyperplane and have the greatest influence on where the hyperplane is located. These vectors are called support vectors because they support the hyperplane. The SVM method assists in identifying the appropriate decision boundary or region, often known as a hyperplane. The SVM algorithm determines which line from each class is closest to the other. Support vectors are the names for these points. Margin is the distance between the hyperplane and the vectors. Maximizing this margin is the aim of SVM. The ideal hyperplane is the one with the largest margin.

6.3.3 Pros and Cons associated with SVM

Pros:
• It performs effectively in high dimensional spaces and has a very obvious margin of separation.
•When the number of samples is less than the number of dimensions, it works well.
•It is also memory efficient because it only needs a small fraction of training points for the decision function (known as support vectors).

Cons:
•Because more training time is needed when we have a large data set, it doesn't perform well.
• It also performs poorly when the data set contains more noise, such as when the target classes overlap.
• Probability estimates are computed via an expensive five-fold cross-validation, not directly provided by SVM. It is a part of the Python scikit-learn library's related SVC algorithm.

6.4 Implementation of support vector machines in Scikit-Learn

I import the train_test_split function from the sklearn library as like logistic regression and split as a 75/25.

from sklearn.model_selection import train_test_split

X_train,X_test,y_train, y_test=train_test_split(x_scaled, y_scaled, test_size=0.30)

Importing SVC (support vector classification) from sklearn.svm is the next step.

from sklearn.svm import SVC
svm_classifier = svm.SVC(kernel = 'linear', gamma='auto', C= 2)

Now fit the X_train and y_train

svm_classifier.fit(X_train, y_train)

yy_prediction = svm_classifier.predict(X_test)

7 Methods

In my project, I had two objectives. Testing machine learning comes first, followed by integrating lora sensors with the gateway. For the machine learning portion, I had around 48 json files which contain different information such as experiment number, box id, length of the files, KMP_start, KMP_stop and temperature value. First I categorized each of the 48 json files as good or bad after that, I extracted the temperature values from this json files. After that, using curve fitting algorithm we calculate different parameters like Pre-exponential factor, Rate constant and Bias. I also calculated Average and Mean Square Error for the model. The dataset was divided into training and testing groups. I created three distinct models using the training dataset, including a neural network model, a support vector machine model, and a logistic regression model.

When the models were ready, I extracted the temperature value and other useful information using python programming language from the json files with the aid of the RN 2483 and Raspberry Pi. To publish into the MQTT Broker, I created a topic and a packet with 10 temperature values and converted them to HEX values. For reduced mistake, I transmitted the packet via OTAA (Over The Air Activation) every ten seconds. I subscribed the topic which was created when it was published. I converted the value from HEX to decimal and stored the data in an array when I received the packet. After receiving all the data, I calculated Pre-exponential factor, Rate constant, Bias, Avg and MSE and used those data to classify the output from the model.

The project's goal is to build an intelligent LoRa gateway based on a Raspberry Pi that can classify the dataset.

7.1 Comparison of different Mathematical Model

This is how a classification accuracy is defined:

$$\text{Accuracy} = \frac{Number\ of\ Correct\ Predictions}{Number\ of\ all\ Predictios} = \frac{TP + TN}{TP + TN + FP + FN}$$

TP = True Positive
TN = True Negative
FP = False Positive
FN = False Negative

A significant issue with utilizing accuracy when evaluating a classification model is when there is an unbalanced distribution of classes within a data set, even though accuracy gives a general idea of the model's correctness in terms of predicting the correct classes. Since, we have unbalance dataset therefore i use confusion matrix to evaluating a model's accuracy. Confusion matrix shows the correct and incorrect predictions on each class. In our project there are two classes that a classification ML model predicted, a confusion matrix shows the true and false predictions on each class. When a logistic regression model is applied to a binary classification problem, the confusion matrix is a 2 by2 matrix. But For N output classification problems, NxN (N by N) matrix will be used. In our project we used binary classification problem therefore, we use 2*2 confusion matrix [13].

Given a 2 by 2 confusion matrix with classes A and B, the terms used inside it are:

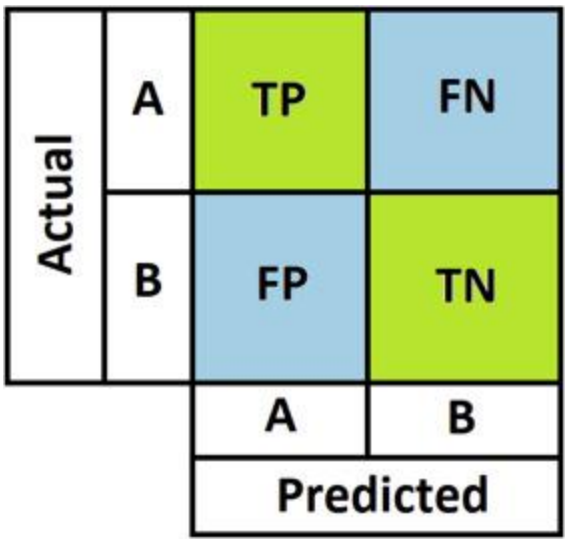

Figure 7.1 Confusion matrix

The comparison of accuracy, precision, and recall for different mathematical models are shown below.

True positive: How often a machine learning model properly classifies a sample as class A.
False negative: how many times a machine learning model misclassifies a sample as class A.
False positive: how many times a machine learning model misclassifies a sample as class B.
True negative: How often a machine learning model properly classifies a sample as class B.

Suppose, we created a classification model to detect a rare disease, such as cancer. A model that predicts all patients are healthy will have 95% accuracy if only 5% of patients have cancer. Even if only 0.1% of patients have cancer, the same model predicts that all patients are healthy with 99.9% accuracy. Of course, the term "accuracy" is deceptive. Both of these models are ineffective for disease detection because they miss all cancers. Other metrics are required to help us weigh the cost of various types of errors.

Precision and recall are metrics used to assess the performance of a classification model. These parameters provide more information about the accuracy of a model. Here is a definition of precision and recall:

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Precision: As shown, precision calculates the accuracy of a model when evaluating positive classes. In other words, out of the predicted positive classes, precision determines the number that is actually positive. Precision is preferred when the cost of false positive is very high.

Recall: Recall measures the number of actual positives. Recall is used when there is a high cost associated with false negative.

As a result, recall and precision are trade-offs. As previously mentioned, precision and recall cannot be maximized at the same time, and depending on the project, one is chosen over the other. These two measurements have an inverse relationship.
In summary:

- Increase precision due to the high cost of false positives.
- False negatives have a high cost, therefore increase recall.

F1 score: Another classification metric that can be applied in cases of unequal class distribution is known as F1 score since it takes both false positive and false negative into account. When a compromise between precision and memory is sought, the F1 score is also helpful. The following is the formula for the F1 score:

F1_score =2 x $\frac{Precision\ x\ Recall}{Precision + Recall}$

8 Results

8.1 Evaluation of Logistic Regression:

To generate the confusion matrix and classification report,We have to import "classification_report" and "confusion_matrix" from "sklearn.metrics" and print the resultant confusion matrix of y_test (actual testing output values) and y_pred (predicted testing output values). In addition, generate a heat map of the resulting confusion matrix.

```
from sklearn.metrics import classification_report,confusion_matrix
print(confusion_matrix(y_test,y_pred))
sns.heatmap(confusion_matrix(y_test,y_pred),
annot=True, cmap="viridis")
```

The confusion matrix and classification report are given below for logistic regression:

```
[[3 0]
 [1 8]]
```

: <AxesSubplot:>

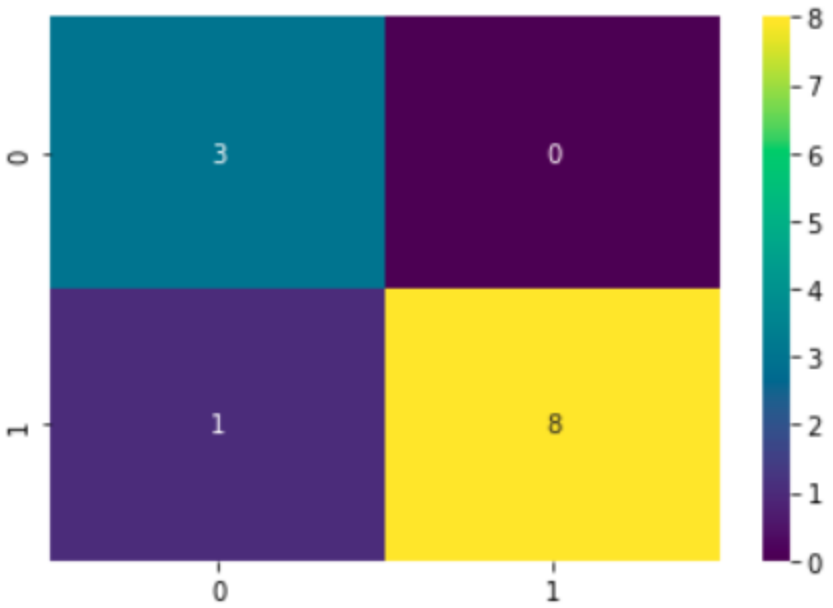

Figure 8.1 Confusion matrix of testing set and its heat map

As seen on the Figure, 3 instances (out of 3) were correctly classified as class 0 and 0 was incorrectly classified under this class. On the other hand, 8 instances (out of 9) were correctly classified as class 1 and only 1 instance was incorrectly classified.

	precision	recall	f1-score	support
0	1.00	0.75	0.86	4
1	0.89	1.00	0.94	8
accuracy			0.92	12
macro avg	0.94	0.88	0.90	12
weighted avg	0.93	0.92	0.91	12

.....

Figure 8.2 Logistic Regression Classification report

The model's total accuracy is 92%. Additionally, class 1 has high precision and recall percentages range from 89% to 100%. We know that when precision is high that means the model avoid a lot of mistakes. On the other hand, For class 0, the precision is high but the recall

is 75% that means it makes more mistake to predict class 0 as class 1.We know that, F1 score is the combination of precision and recall. For class 1 the F1 score (94%) is high compare to class 0 (86%)

8.2 Evaluation of Support Vector Machine:

We'll take the same actions we did before to make the comparison fair. We will import "classification_report" and "confusion_matrix" from "sklearn.metrics" and display the resultant confusion matrix of y_test (testing values) and y_pred (predicted testing values). In addition, I generated a heat map of the resulting confusion matrix.

```
..................................................
[[3 0]
 [3 6]]

Out[17]: <AxesSubplot:>
```

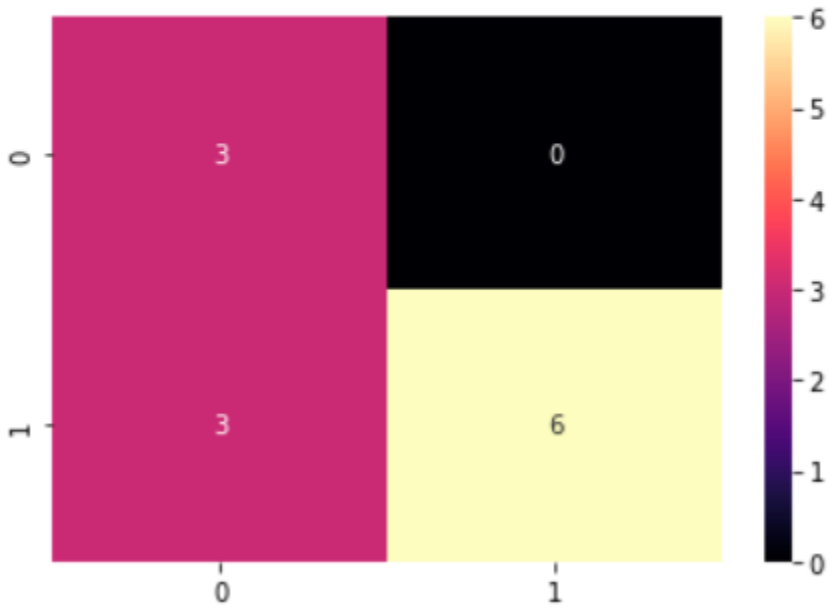

Figure 8.3 Confusion matrix of testing set and its heat map

As can be observed from the Figure, 0 instances was wrongly classified under this class 0 while 3 instances (out of 3) were correctly classified as class 0. On the other hand, 3 instance was wrongly classified, leaving 6 out of 9 examples to be appropriately classified as class 1.

	precision	recall	f1-score	support
0	1.00	0.50	0.67	6
1	0.67	1.00	0.80	6
accuracy			0.75	12
macro avg	0.83	0.75	0.73	12
weighted avg	0.83	0.75	0.73	12

.....

Figure 8.4 Support Vector Machine Classification report.

The accuracy for Support Vector Machine is less than Logistic Regression. The overall accuracy of the model is 75%. Furthermore, the class 0 precision is 100% that means it does not make any mistake to predict class 0 where the precision is 67% for class 1. In contrast, class 1 recall is 100% where the percentage of recall is only 50% for class 0.
The F1 score is also low compare to Logistic Regression in both class 0 and class 1. The F1 score for class 0 is 67% where 80% for class 1. I also verified that changing the kernel changes the accuracy. When I change the kernel to "rbf"(radial basis function) then accuracy is 67% on the other hand when the kernel is selected to "poly"(Polynomial) then the accuracy (92%) is increasing which is equivalent to logistic regression.

8.3 Evaluation of Neural Network Model:

I repeated my previous action to ensure a fair comparison. I imported the "classification_report" and "confusion_matrix" from "sklearn.metrics" library and print the of resultant confusion matrix for y_test (testing values) and y_pred (predicted testing values). In addition, i generated a heat map of the resulting confusion matrix as like before. The accuracy depends on number of epoch and learning rate. Here we use learning rate 0.01.In my project, i trained the model with 500 epochs and 2000 epochs. The results of the models are given below

The confusion matrix for 500 epochs is given below:

```
....................................
[[3 0]
 [2 7]]

Out[34]: <AxesSubplot:>
```

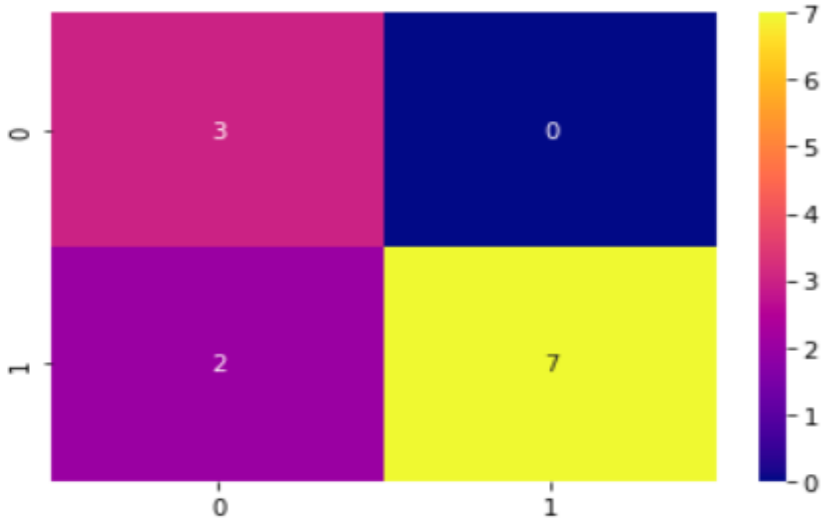

Figure 8.5 Confusion matrix of testing set and its heat map.

As can be seen from the Figure, there were 3 instances (out of 3) which were correctly classified as class 0 and 0 was incorrectly classified under this class. Meanwhile, there were7 instances (out of 9) were correctly classified as class 1 and only 2 instances were incorrectly classified.

	precision	recall	f1-score	support
0.0	1.00	0.60	0.75	5
1.0	0.78	1.00	0.88	7
accuracy			0.83	12
macro avg	0.89	0.80	0.81	12
weighted avg	0.87	0.83	0.82	12

```
....................................
```

F [6]igure 8.6 . Neural Network Classification report for 500 epochs.

From the above classification report we have seen that the precision is 100% for class 0 and 78% for class 1 similarly, the recall is 100% for class 1 and 60% for class 0. The accuracy of the model is 83% which is lower than logistic regression model but higher than support vector

machine model. The F1 score is also low in both classes which is 75% and 88% respectively.

Here, the confusion matrix for 2000 epochs is given below:

```
....................................
[[3 0]
 [1 8]]

Out[25]: <AxesSubplot:>
```

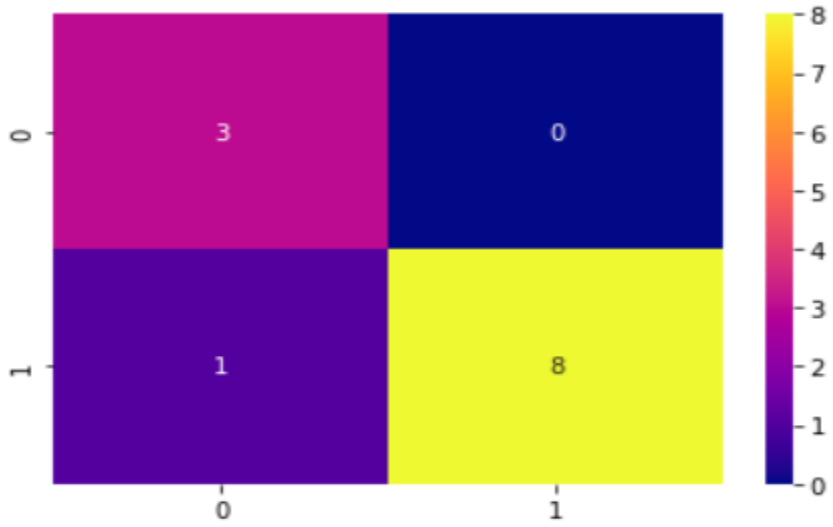

Figure 8.7 Confusion matrix of testing set and its heat map

From the confusion matrix table, we have seen that, this confusion matrix is similar to logistic regression confusion matrix. 3 instances (out of 3) were correctly classified as class 0 and 0 was incorrectly classified under this class. On the other hand, 8 instances (out of 9) were correctly classified as class 1 and only 1 instance was incorrectly classified.

	precision	recall	f1-score	support
0.0	1.00	0.75	0.86	4
1.0	0.89	1.00	0.94	8
accuracy			0.92	12
macro avg	0.94	0.88	0.90	12
weighted avg	0.93	0.92	0.91	12

```
....................................
```

Figure 8.8 Neural Network Classification report for 2000 epochs.

The overall accuracy of the model after 2000 epochs is 92% which is similar to Logistic regression model. The accuracy will be higher if we increase the number of epochs. From the classification report we have found that the precision for class 0 is 100% which is maximum but for class 1 is 89%. If we check the recall from the classification report we found that class 1 recall is 100% where class 0 recall is 75%. The F1 score is also high (94%) for the class 1 compare to class 0(86%).

```
epoch: 100, loss = 0.5745
epoch: 200, loss = 0.4857
epoch: 300, loss = 0.4320
epoch: 400, loss = 0.3949
epoch: 500, loss = 0.3672
acc: 0.8333333333333334
```

```
epoch: 1500, loss = 0.2643
epoch: 1600, loss = 0.2592
epoch: 1700, loss = 0.2547
epoch: 1800, loss = 0.2505
epoch: 1900, loss = 0.2467
epoch: 2000, loss = 0.2432
acc: 0.9166666666666666
```

Figure 8.9 Deep Neural Network losses

As can be seen from the table above, the losses are reducing for both scenarios as the number of epochs rises. I've also seen that the accuracy remains at 92% even after increasing the epochs by 5000 or 10000.

9 Problems During Implementation

During my project implementation, I faced some problems which are discussed below.

I found that if we send the lora packet in short interval of time (<10second) then we will lose more packet on the receiver. Sometimes I have fount that the channel is busy so I have to send the data again.

MQTT is a trustworthy solution, but there are also certain issues that come with it. In MQTT, if the incoming event message length is longer than 256 byte then we cannot send all the message. That means, this may have an effect on systems that send large amounts of data. But MQTT appears to operate quite quickly when transferring small amounts of data. I also faced a problem to connect the Gateway with the TTN (The Things Networks) because of the Gateway EUI. When I sent the data over the ABP (Activation By Personalization) there was more error compare to OTAA(Over the Air Activation).

Due to the interference, when I extracted the sensor data from our gateway, I also received the other sensor data. Similarly, when I publish sensor data, it passes through three to four gateways.

I intended to develop a system that can keep track of a device's condition and respond appropriately. I required the integration of various models as well as an open-source solution. I was successful in achieving all of my objectives.

10 Conclusion

We can infer from the above tables that the Logistic Regression provides the greater accuracy which is 92%. We will achieve greater accuracy for Neural Network but Neural Networks are more effective for big data but Logistic Regression is computational more efficient than Neural Network therefore, I use logistic regression in my project. Python is one of the best programming languages, and I use it in my project. The python interface works best and has significantly lesser delay than other interfaces.

11 Future research

We intend to work with real-time data from temperature and humidity sensors in the near future. Theoretically, we can connect thousands of sensors, but in practice, we'll see how many sensors we can actually connect to the gateway and how well it performs. In the future, we can reduce the publish interval time and check the efficiency.

12 Appendix A

12.1 Subscribe the data from TTN server

The code below shows how MQTT Subscriber receives temperature data from TTN server.

mqtt_project_selim.py

To subscribe the sensor data from TTN a server we need the username and password which one we get from the TTN website.

Username = "dragino-sensors-rj@ttn"

Password="NNSXS.OYYME4TUAIQI2VUYV5SEFFEHXVV6V64DTC75X4Y.VJX2GZDI24RZJNWZTHJL6S2ZKAO3CFO7722FQMPV2RWG4KORUBVQ"

After receiving the json format data, separate and decode the temperature data by using the following code.

```
data = json.loads(msg.payload.decode('utf-8'))
text = data['uplink_message']['frm_payload']
 zz = base64.b64decode(text) # decode the payload
 zzz = zz.decode('ascii')
```

After decoding all the data we send those values to the find_result function and this function gives the desired output.

```
result = find_result(my_data_lst)
print(result)
```

12.2 Publish the data to TTN Server

The code below demonstrates how to run the program using Argument Parser. After that, the data is read Serially from the json files and send this data to TTN server every 10 seconds.

```
parser = argparse.ArgumentParser(description='Enter the Sensor and Box nuumber')
ser = serial.Serial(port, 115200, timeout=0.050)
# to join over the air activation
command = "mac join otaa" + "\r" + "\n"
ser.write(command.encode('utf-8'))
my_hex = my_val.encode("utf-8").hex()
command = "mac tx uncnf 1 " + my_hex + "\r" + "\n"
ser.write(command.encode('utf-8'))
```

12.3 Logistic Regression Model

This code shows the logistic regression model. Extract the data from the csv files and labelled them to “0” and “1” for the machine learning. Then, split the data into training data and testing data. After training the model we will write this model using pickle library.

```
#read the csv file
```

```
df = pd.read_csv('my_project_data.csv')
Map this label
#mapping
df['label']=df['label'].map({'bad':0,'good':1})
#train-test split
X_train, X_test, y_train, y_test = train_test_split(
   X, y, test_size=0.25, random_state=40)
#make a model
classifier=LogisticRegression()
```

12.4 Necessary Python Library for the Project

In my project, I had used different python function from different python library. The list of the libraries are given below.

- Paho.mqtt
- Json
- Pandas
- Numpy
- Scipy
- Serial
- Argparse
- Os
- Sklearn
- Pickle

13 Bibliography

[1] B. Paul, "An Overview of LoRaWAN," 2020.

[2] L. D. S, "Bevywise," 12 05 2021. [Online]. Available: https://www.bevywise.com/blog/author/lakshmi-deepa-s/.

[3] Techplayon, "Network and Protocol Architecture & Frame Structure," 10 2018. [Online]. Available: https://www.techplayon.com/lora-long-range-network-architecture-protocol-architecture-and-frame-formats/.

[4] K. Manditereza, "HIVEMQ," 14 12 2021. [Online]. Available: https://www.hivemq.com/blog/iot-event-driven-microservices-architecture-mqtt/.

[5] TIBCO, "TIBCO Software," [Online]. Available: https://www.tibco.com/reference-center/what-is-event-driven-architecture.

[6] K. Singh, "Connecting Live Sensor Data with Digital Twins and," Bremen, 2022.

[7] Behr Technologies Inc., "Behrtech," [Online]. Available: https://behrtech.com/blog/mqtt-in-the-iot-architecture/.

[8] Comparitech Limited, "comparitech," 2015. [Online]. Available: https://www.comparitech.com/net-admin/what-is-mqtt/.

[9] Raspberry Pi Foundation, "Raspberry Pi," [Online]. Available: https://static.raspberrypi.org/files/product-briefs/Raspberry-Pi-Model-Bplus-Product-Brief.pdf.

[10] J. Brownlee, "Mechine Learning Mastery," [Online]. Available: https://machinelearningmastery.com/curve-fitting-with-python/.

[11] P. Antoniadis, "Baeldung," 07 02 2022. [Online]. Available: https://www.baeldung.com/cs/curve-fitting.

[12] N. Venkatesan, "Towards Data Science," [Online]. Available: https://towardsdatascience.com/basic-curve-fitting-of-scientific-data-with-python-9592244a2509.

[13] A. H. Hoss Belyadi, Machine Learning Guide for Oil and Gas Using Python, 2021.

[14] A. Minteer, "O'Reilly," [Online]. Available: https://www.oreilly.com/library/view/analytics-for-the/9781787120730/db65d957-cf17-459c-a203-4b8234a14261.xhtml.

[15] X. Mesrobian, "Automation World," [Online]. Available: https://www.automationworld.com/factory/iiot/article/22235994/advantages-of-a-smart-mqtt-broker.

[16] Rf Wirless World, "Rf Wirless World," [Online]. Available: https://www.rfwireless-world.com/Terminology/Advantages-and-Disadvantages-of-MQTT.html.

[17] Rajiv, "RF Page," 31 07 2022. [Online]. Available: https://www.rfpage.com/applications-future-lora-wan-technology/.