2019

University of North Carolina Wilmington

Master of Science in

Computer Science and Information Systems

Proceedings

https://csbapp.uncw.edu/mscsis

MACHINE LEARNING FOR INTERNET OF THINGS

Predict Refrigerator Performance Using Temperature Data

Adviser: Dr. Clayton Ferner
Committee: Dr. Judith Gebauer
Dr. Curry Guinn

DHAVAL CHAUHAN

MS – CSIS

dnc2848@uncw.edu

Table of Contents

List of Figures 3
List of Tables 5
Abstract 6
1 Introduction 7
2 Internet of Things 9
2.1 User interface 9
2.1 Cloud 10
2.2 Gateway 10
2.3 Sensor 10
3 Wireless sensor network 12
3.1 WSN topologies 12
3.1.1 Star topology 13
3.1.2 Tree 13
3.1.3 Mesh 13
3.2 The components of a sensor node 13
3.3 Operating Systems and protocols 14
3.3.1 Zigbee 14
3.3.2 6loWPAN 15
4 Machine learning 16
4.1 Supervised learning 17
4.2 Decision Tree 17
4.3 Regression and Classification 17
4.4 Instance Based Learning 18
4.5 Ensemble Learning 19
4.6 Unsupervised Learning 19
5 RNN 20
5.1 Introduction 20
5.2 LSTM 21
6 Data collection 23
6.1 Tools and Software 24
7 Regression Model Results 25
7.1 Polynomial Regression using Numpy 25

7.2 Support Vector Regression 27
8 LSTM Model Results 29
8.1 Root Mean Square Error (RMSE) Comparison 29
8.2 Neuron count comparison 30
8.3 Training & Testing with model of another sensor 33
8.4 Temperature Prediction – Sensor Groups 36
8.5 Two Model Approach 40
8.6 Temperature and Failure prediction before and after failure 42
8.6.1 Sensor 1 Results 44
8.6.2 Sensor 2 Results 47
8.6.3 Sensor 3 Results 50
8.6.4 Sensor 4 Results 53
8.6.5 Sensor 5 Results 56
8.6.6 Sensor 6 Results 59
9 Conclusion 62
10 Future Work 63
11 References 65

List of Figures

Figure 1 Internet of things architecture 9
Figure 2 Verisolutions Inc. Gateway 11
Figure 3 Verisolutions Inc. Sensor 11
Figure 4 WSN topologies 12
Figure 5 Frequency Distribution 14
Figure 6 Traditional program 16
Figure 7 Machine Learning 16
Figure 8 Polynomial Regression 18
Figure 9 Standard RNN 21
Figure 10 LSTM 22
Figure 11 Symbols 22
Figure 12 Raw data in excel file 23
Figure 13 Polynomial Regression (Degree 20) 25
Figure 14 Polynomial Regression (Degree 35) 26
Figure 15 Polynomial Regression (Degree 45) 27
Figure 16 RBF Model vs Raw Data 28
Figure 17 RMSE Comparison for different Input Data Length in Days 29
Figure 18 Real vs Prediction at different Neuron count 30
Figure 19 Underfitting 31
Figure 20 Overfitting 32
Figure 21 Best Fit (NN 16) 32
Figure 22 Train Sensor 3 - Test Sensor 3 34
Figure 23 Train Sensor 1 - Test Sensor 3 34
Figure 24 Train Sensor 2 - Test Sensor 3 35
Figure 25 Train Sensor 4 - Test Sensor 3 35
Figure 26 Train Sensor 5 - Test Sensor 3 36
Figure 27 SeaFood Sensor 1 Temperature 37
Figure 28 SeaFood Sensor 2 Temperature Prediction 38
Figure 29 SeaFood Sensor 3 Temperature Prediction 38
Figure 30 SeaFood Sensor 4 Temperature Prediction 39
Figure 31 SeaFood Sensor 5 Temperature Predictions 39
Figure 32 Sensor 1 Temperature vs Failure Data 44
Figure 33 Sensor 1 Temperature Prediction before failure 44
Figure 34 Sensor 1 Failure Prediction before failure 45
Figure 35 Sensor 1 Temperature Prediction after failure 45
Figure 36 Sensor 1 Failure Prediction after failure 46
Figure 37 Sensor 2 Temperature vs Failure Data 47
Figure 38 Sensor 2 Temperature Prediction before failure 47
Figure 39 Sensor 2 Failure Prediction before failure 48
Figure 40 Sensor 2 Temperature Prediction after failure 48
Figure 41 Sensor 2 Failure Prediction after failure 49

Figure 42 Sensor 3 Temperature vs Failure 50
Figure 43 Sensor 3 Temperature prediction before failure 50
Figure 44 Sensor 3 Failure prediction before failure 51
Figure 45 Sensor 3 Temperature prediction after failure 51
Figure 46 Sensor 3 Failure prediction after failure 52
Figure 47 Sensor 4 Temperature vs Failure 53
Figure 48 Sensor 4 Temperature prediction before failure 53
Figure 49 Sensor 4 Failure prediction before failure 54
Figure 50 Sensor 4 Temperature prediction after failure 54
Figure 51 Sensor 4 Failure prediction after failure 55
Figure 52 Sensor 5 Temperature vs Failure Data 56
Figure 53 Sensor 5 Temperature prediction before failure 56
Figure 54 Sensor 5 Failure prediction before failure 57
Figure 55 Sensor 5 Temperature prediction after failure 57
Figure 56 Sensor 5 Temperature prediction after failure 58
Figure 57 Sensor 6 Temperature vs Failure 59
Figure 58 Sensor 6 Temperature prediction before failure 59
Figure 59 Sensor 6 Failure prediction before failure 60
Figure 60 Sensor 6 Temperature prediction after failure 60
Figure 61 Sensor 6 Failure prediction after failure 61

List of Tables

Table 1 RMSE value for figure 17 31
Table 2 SeaFood Cooler Group Temperature Profile 37
Table 3 Raw data split 41
Table 4 Temperature Prediction 42
Table 5 Failure prediction 42

Abstract

This paper discusses the architecture of *Internet of Things* and how Machine Learning can help to predict refrigerator temperature and performance. *Internet of Things* (IoT) is a term used for a system which can collect data from various sources, such as lights, refrigerators, coolers, soil, water, air, etc. Sensors collect data and publish it to the cloud. This document explains methods to use data and analyze it to predict the behavior of the system attached to the sensors. The sensors designed by Verisolutions Inc. Atlanta are used to collect refrigerator data. Temperature and humidity data collected from the sensors is used for training the machine learning algorithm. As per the USDA guidelines, refrigerators should maintain a temperature of 40 °F or below. When the refrigeration unit fails, the temperature inside rises in 3-4 hours. The machine learning algorithm can find a pattern and notify users with the potential failure alert.

1 Introduction

The Internet of Things (IoT) is a new technology that can impact how we do our daily tasks and how we live. It connects devices to the Internet by integrating micro-controllers with systems, such as refrigerators, lighting equipment, ovens, doors, windows etc. However, the main goal is to collect data and derive useful information from such systems, which can be monitored and controlled through the Internet.

Artificial Intelligence can improve the quality of the system by interpreting the huge amount of data collected by the sensors. Moreover, a *Machine Learning* (ML) algorithm can make data collected by sensors better suited for analytics. A business can use various ML algorithms to train data and test data. It can provide predictions, potential actions, or recommendations. This approach can make predictive maintenance more effective than periodic inspection by the human. There are many examples such as thermostat control, weather prediction, air quality prediction, and refrigerator performance prediction where ML can provide deep insight into the performance of the system. In this project, machine learning techniques are applied to data collected from sensors installed in refrigerator units in an attempt to predict failure.

Temperature and humidity sensors can be placed inside a refrigerator to collect *real time* data. Users can monitor the data to determine if the refrigerator is working properly. This system can help to keep food safe at a specific temperature. An ML algorithm applied on data collected by a sensor can provide an early alert of equipment maintenance requirement or failure. A business can save money and inventory by addressing a failing refrigerator unit before failure.

Restaurants must keep various food items at a specific temperature to keep it fresh. If the temperature is not maintained for a period of time, food is no longer safe to eat. At present, most restaurants monitor the temperature manually every few hours, which is a procedure with the potential of human error that can lead to inventory loss. This problem is the motivation behind the current project: to propose a solution to alert users of a potential failure in advance using historical data.

This paper explains the architecture of *Wireless Sensor Network* (WSN) and *Internet of Things* (IoT). Data is collected using sensors designed by *Verisolutions Inc.,* a company that, provides temperature and humidity monitoring solutions to restaurants. In this project, various development strategies are attempted to find a better solution than a purely manual process. At first, linear regression and polynomial regression models are used to find a temperature pattern in the collected data. In addition, *Support Vector Regression* (SVR) model is considered as another solution of regression. After this, Machine Learning techniques are explored to predict temperature and failure of the refrigerator. There are many ways of using supervised and un-supervised learning to train the model. In this project, *Long Short Term Memory* (LSTM) is used to train the model.

This paper is organized as follows: First, I will discuss the architecture of the Internet of Things technology and the component of a Wireless Sensor Network. Second, I will introduce Machine Learning and discuss the techniques used in this project. Third, I will discuss the data collection and tools used in this project. Next, I will discuss the application of Machine Learning techniques to the data collected as I attempt to find the best model to predict refrigerator performance. After that, I explore whether these techniques can obtain the ultimate goal of predicting refrigerator failure before it happens. Finally, I will conclude and provide some suggestions for future work.

2 Internet of Things

The Internet of Things is a fast-growing technology. Until recently, it was used mostly to monitor various types of electromechanical systems. Smart Home and Smart City initiatives taken by many countries are expected to add billions of devices to the Internet soon, ranging from common household items like coffee makers and refrigerators, to a home's indoor and outdoor lighting systems. A smart device connected to the Internet attempts to learn its user's patterns and habits. With these innovations, a smart device can potentially respond accordingly through learning. It has been predicted that a massive 75 billion devices will be connected to the internet by 2020. (Next Big Things in IoT predictions for 2020, n.d.)

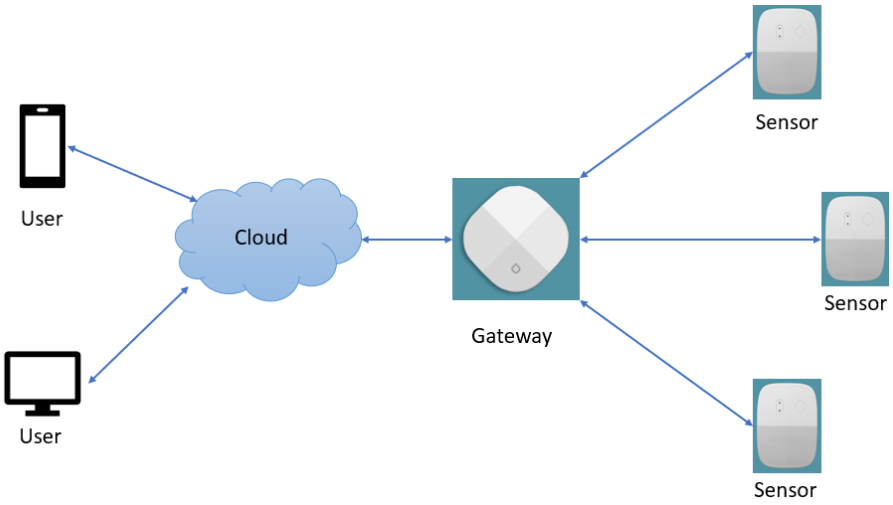


Figure 1 Internet of things architecture

An Internet of Things system contains four major components (Figure 1):

1. User interface
2. Cloud
3. Gateway
4. Sensor

2.1 User interface

The users can use different methods to see and understand the data collected by sensors. It can be anything from software running on a computer to Android/iOS applications installed on smart phone

or to the voice and speech-controlled Amazon Alexa or Google home devices. Native apps for smartphone can be downloaded easily to a smart phone to be used for configuring systems, managing accounts or getting alerts. The type and skill level of users also impacts the design strategy of UI. For example, customers in a restaurant typically prefer an easy-to-use UI for temperature monitoring, while customers in healthcare might prefer detailed and analytics-type UI.

2.1 Cloud

Sensors generates large amounts of data every day. All data has to be stored and prepared for generating analytics for users. Data can be stored in many ways and can then be used for processing. One of the most common ways is to use a third-party database solution or the cloud, such as Amazon Web Services (AWS) provided by database companies. The cloud includes the web-services used to create daily or monthly reports of a sensor's performance and usage. It can generate analytics of the data collected by the sensor.

2.2 Gateway

The Gateway is a physical electronic device which serves as a link between the cloud and sensors. A gateway is the most powerful piece of hardware in an IoT system. It can process some data locally before sending it to the cloud. It can protect the data being sent to the cloud by adding additional security features. A Gateway is usually powered on all the time to establish a robust connection between the cloud and sensors. It can send data to the cloud by various ways like cellular connection, Wi-Fi, or Bluetooth. It also serves as a main server for all sensors in a wireless sensor network. It must operate on the same RF frequency as the sensors to allow WSN operations.

2.3 Sensor

A sensor is an electro-mechanical device which is equipped with a microcontroller and other supporting components such as a battery, crystal oscillator, voltage regulator, etc. One or many sensors can communicate with the gateway depending on the Wireless Sensor Network (WSN) topology (Figure 2 and 3). A sensor is powered by batteries so that it can be placed where needed to measure various things like temperature, humidity, pressure, etc.

Figure 2 Verisolutions Inc. Gateway

Figure 3 Verisolutions Inc. Sensor

3 Wireless sensor network

A *wireless sensor network* (WSN) is a network with distributed sensors which is used to monitor various types of parameters like temperature, humidity, pressure, luminance, etc. A gateway provides wired connectivity to the Internet and the rest of the sensor network. Many wireless protocols such as ZigBee, 6loWPAN, LoRa are available to design a system which depends on the requirements and the limitations of the sensors. The sensors use radio frequencies (RF) to send data to the gateway. The selection of a proper radio frequency is a very important aspect of any WSN. The data can be transmitted over long distance by selecting lower frequency while the higher frequency data is only useful for short distance transmission.

3.1 WSN topologies

WSN nodes connected with the gateway can be arranged in a variety of topologies (Figure 4). The three most commonly used topologies are:

1. Star
2. Tree
3. Mesh

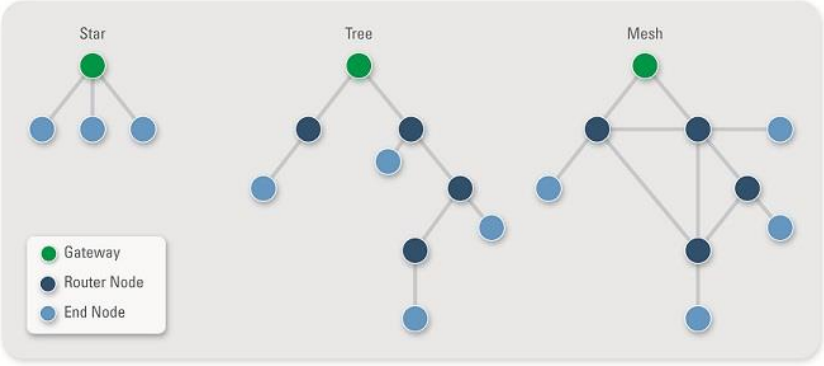


Figure 4 WSN topologies

(Instrument, 2012)

3.1.1 Star topology

In a Star topology, all sensor nodes communicate with the gateway directly with a one-to-one link. This topology is centered around a gateway. If the gateway stops working, the entire network stops working; however, the sensors are not dependent on each other for sending data to the gateway.

3.1.2 Tree

In a tree topology, the gateway creates a network as the root node. All other nodes join this network by connecting to one other node as its parent with network joining messages. A parent will forward messages of its child to its parent. This parent-child relationship is dynamic. If the gateway is far from the sensor node, then the sensor can select any other nearby sensor as its parent. This feature increases the overall range of the sensor network.

3.1.3 Mesh

In a Mesh topology, all sensor nodes send messages to each other. All messages will eventually reach the gateway. Messages die after a certain amount of time, known as the time-to-live. This topology has higher RF traffic compared to Tree and Star but has more chances of guaranteed delivery of messages.

3.2 The components of a sensor node

Every WSN sensor node is consist of several electronic components. The selection of the components depends on size of sensor, frequency of operation, projected lifetime etc. Some of the major components are:

1. Microcontroller unit (MCU)
2. Battery
3. Printed circuit board (PCB)
4. Sensor interface
5. Antenna

The sensor can be powered by a power adapter or battery. A power adapter provides constant voltage to the microcontroller unit (MCU), whereas a battery's power output is limited. The power consumption of any sensor depends on the hardware used to design the PCB and voltage regulator analog circuit. The data collection interval and operating system can consume considerable amount of energy. It is important to use proper protocols and operating system to control power consumption. The weight and size of the components is also important to sensor design.

Wireless sensor network nodes can use various radio frequencies (RF) to communicate with each other. As of now there are two RF bands that are popular for WSN systems:

1. 2.4 GHz
2. Sub GHz

The 2.4 GHz band has 15 unique channels with different frequencies. As shown in Figure 5, a sensor network can select any frequency within 2405 MHz to 2470 MHz with the IEEE 802.15.4 protocol. The IEEE 802.11 protocol is used for Wi-Fi/WLAN communication over 2.4 GHz frequency.

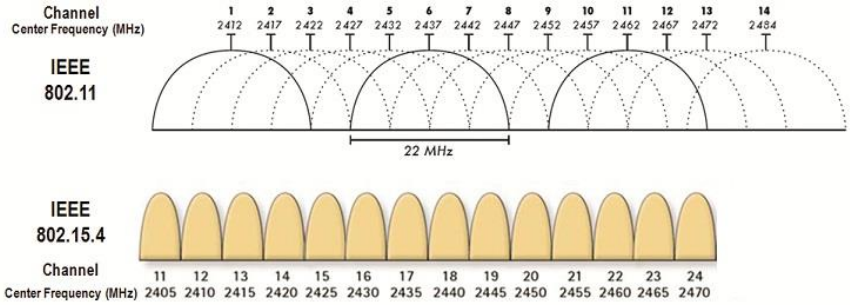


Figure 5 Frequency Distribution

(Instrument, 2012)

The Sub 1GHz radio frequency operates in the *Industrial, Scientific and Medical* (ISM) band. It can select any frequency between 769-935 MHz, 315 MHz and 468 MHz. By selecting lower frequencies, the sensor module can send data over long distances such as a mile. However, the data transmission rate is lower with lower frequency.

The size of the antenna depends on the operating frequency of the device. An antenna is a core component of any communication system. The purpose of the antenna is to convert an RF signal into electromagnetic waves which can travel in free space. An antenna maintains the same characteristic while sending or receiving data. An antenna must be tuned to the same frequency as transmitter and receiver. The antenna emits the radiation in different patterns which is called radiation pattern. It is used to describe the strength of a radiation field in different directions.

3.3 Operating Systems and protocols

3.3.1 Zigbee

The ZigBee Alliance is an association of companies working together to develop standards for reliable, cost-effective, low-power wireless networking (alliance, 2012) . ZigBee technology is used in a wide range of products for consumer electronics and industrial systems. ZigBee is built using the standard IEEE 802.15.4 protocol. It provides a framework to develop a wireless network on top of Physical RF and MAC layer with 8-Byte MAC address.

The physical layer supports three radio frequency bands.

1. 2.4 GHz
2. 868 MHz
3. 919 MHz

The MAC layer has two types of devices.

1. FFD (Full Function Device)
2. RFD (Reduced Function Device)

The FFD can work as both a coordinator and end-device, while RFD can only function as an end-device. All end-devices must be connected with the coordinator to form a network. The Network layer and application layer protocols are used to form a network to send useful data like sensor details and sensor configuration.

3.3.2 6loWPAN

The 6loWPAN protocol is built on top of the IEEE 802.15.4 protocol. It provides IPv6 connectivity to sensor nodes. The physical and MAC layers of 6loWPAN are based on the IEEE 802.15.4 protocol. A 6loWPAN network has three types of devices.

1. Server
2. Router
3. End-device

The Server is responsible for creating the IPv6 network. The router and end-device get unique IPv6 addresses from the server when they join the network. Network joining, and maintenance are done via the transport layer protocol RPL. RPL protocol is dynamic in nature. It can make, and brake chains based on the signal strength of the RF. The 6loWPAN protocol follows the tree topology. A server maintains a parent-child table to forward messages to any node. Each node is accessible via a unique IPv6 address.

4 Machine learning

In this section, I introduce *Machine Learning (ML)* and some of the techniques used in ML. Machine Learning refers to computing systems that provide data analytics and predictions. It can also be described as a method of turning data into software. (Barnes, 2015)

Traditional computer programs are used to generate output from a given input. However, in Machine Learning, input data and output are given to find a relationship between them. In other words, a Machine Learning algorithm is used to find a program between given inputs (Figure 6 and 7).

Traditional Program

Figure 6 Traditional program

(Barnes, 2015)

Machine Learning

Figure 7 Machine Learning

(Barnes, 2015)

One ML algorithm cannot work best for all types of problems. Size, variation, and structure of data affects the performance of any ML algorithm. The selection of the best algorithm requires testing many algorithms with the same data and then comparing the results. Each algorithm has its own strengths and weaknesses. The ML algorithms are classified into two major groups:

1. Supervised Learning
2. Unsupervised Learning

4.1 Supervised learning

This algorithm uses independent variables and dependent variables (targets). A dependent variable is predicted by applying various algorithms to the input variable. A Machine Learning algorithm attempts to find a pattern between sequences of input data. The training process of the model depends on the nature of input and output/target data. Examples include; Decision Tree, Regression, and K-Nearest Neighbor (KNN) algorithm.

4.2 Decision Tree

A Decision tree covers both classification and regression. It can be used for a visual representation of decisions. It can be used for both data mining and machine learning. The classification tree is used to classify different entities into groups. Conversely, the regression tree is used for time series analysis and prediction of continued data.

4.3 Regression and Classification

Linear Regression

A Linear Regression is the most used and well-known algorithm for Machine Learning and statistical analysis. Linear regression can be considered a linear model which tries to find a linear relationship between one or many input variables and a single output (dependent) variable. The value of a dependent variable can be calculated by using the *co-efficients* and the independent variables of the linear regression equation.

A simple example of linear model is:

$$Y=AX + C$$

Where,

Y - output variable

X - input variable

A and *C* - Co-efficients

Polynomial Regression

A Polynomial Regression is used where the relationship between input and output variable is non-linear. Linear regression is a very poor choice of a model if the data is not in linear shape. Polynomial regression

is used to capture the curve of the data. As shown in Figure 8, an underfit model does not change significantly with changes to the independent variable. For this specific set of data, a polynomial of degree of 3 can be considered a correct fit. In contrast, a polynomial of degree of 20 can fit all the values but has too much variation to be useful as a predictor. This is an example of overfitting the data. The prediction error can be broken down into two parts:

1. Bias Error
2. Variance

The bias error is a model's assumption to fit the target function and the data. High bias can be considered as a failure by the algorithm to find the relationship between inputs and target outputs. The variance is the amount of the sensitivity to variations in the data. High variance can result in an algorithm modeling the noise instead of the intended output data (overfitting).

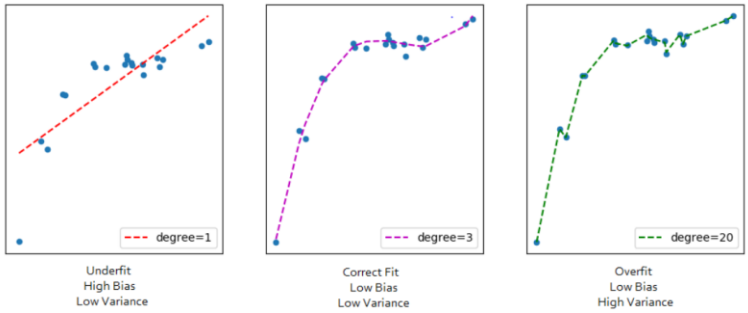


Figure 8 Polynomial Regression

(Polynomial Regression, n.d.)

4.4 Instance Based Learning

With *Instance Based Learning*, inputs are stored locally. When new input arrives, a new query is generated where new input/examples are compared with the stored examples. The target function searches the new input in the database to find the nearest output.

K-Nearest Neighbor (KNN)

The *K-Nearest Neighbor* (KNN) algorithm is used for both regression and classification. The KNN method assumes all instances or input data correspond to points in the n-dimensional space. The nearest neighbors of an instance are defined in terms of the standard Euclidean distance. (K-Nearest Neighbor, n.d.)

4.5 Ensemble Learning

Ensemble learning is used to improve the performance of machine learning by combining more than one model. This method may provide better prediction of data compared to prediction with one model. Ensemble learning can be considered as a meta-algorithm. The combined ML technique can decrease variance (bagging), bias (boosting), and improve predictions. In bagging, random samples of the training data are created. The next step is to build a classifier for each sample. At last, the results of multiple classifiers are combined. Boosting can be considered as a sequential learning of the predictors. At first, a predictor uses the whole data set for learning. After that, a subsequent predictor uses the training data set based on the performance of the previous predictor.

4.6 Unsupervised Learning

With unsupervised learning there is no output or target variable to predict. It is used for clustering data into different groups based on their structure and then to draw inferences from datasets consisting of input data without labeled responses. (Unsupervised Learning, n.d.) It is used to find hidden patterns inside data.

5 RNN

5.1 Introduction

Recursive neural networks (RNN) are a class of neural networks which are used to find the sequential nature of the input. It is widely popular for time series analysis as well as speech and text recognition. For example, it can predict the next word in the sentence "The weather is ____." as "*hot"* or *"cold".*

With the basic RNN, *neuron-like* nodes are organized into layers to create a network. Each node in a given layer is connected to every node in the next layer. An RNN can map the entire history of past inputs to outputs. The connections in an RNN preserves the memory of previous inputs in the internal states of the network to predict the network output. The same set of weights are applied iteratively over a prepared set of inputs to generate the known output. The neural network can be described as a combination of the following three types of layers:

1. Input Layer
2. Hidden Layer
3. Output Layer

A neural network's capability and shape can be described by 5 different parameters.

1. Size – Number of nodes in the model
2. Depth – The number of layers in a network
3. Width – Node count within a specific layer
4. Capacity – Type or structure of the functions computed by each node
5. Architecture – Arrangement of the layers

An RNN can be thought of as a graph of RNN cells, where each cell performs the same operation on every element in the sequence. (Antonio Gulli, 2017) It can be used to solve different types of problems by changing the arrangement in the graph. There are two variants of RNN which work well for time series problems: Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU). The LSTM uses 3 gates to process the data.

1. Forget gate
2. Update gate
3. Output gate

The GRU uses only 2 gates to process the data.

1. Update gate
2. Reset gate

5.2 LSTM

Long Short Term Memory networks (*LSTMs*) are a special variant of RNN, which are capable of learning long-term dependencies. The theory was developed by Hochreiter and Schmidhuber in 1997. LSTMs works well on a wide variety of problems.

LSTM networks use chains of repeating modules of neural networks. The standard RNNs uses a single *tanh* layer in a simple structure (Figure 9). In LSTM, tanh is an activation function. The activation function is used to determine the output of a neural network like 1 or 0. It is used to map the resulting values between -1 to 1 or 0 to 1 etc. The range of tanh function is from -1 to 1.

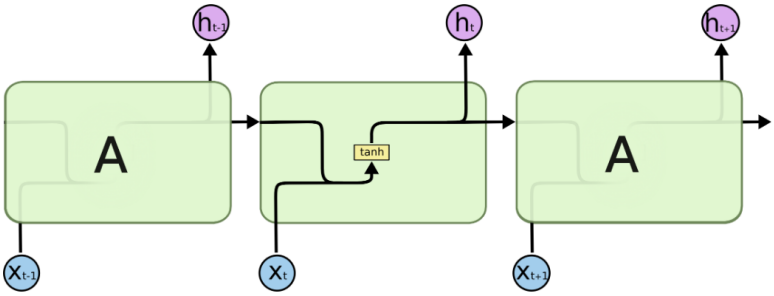


Figure 9 Standard RNN

(Understanding LSTM Networks, 2015)

In the LSTMs, repeating modules have different structures. Four different neural network layers interact in every module (Figure 10).

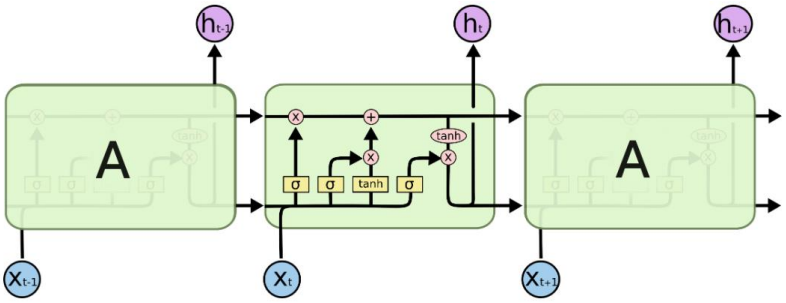


Figure 10 LSTM

(Understanding LSTM Networks, 2015)

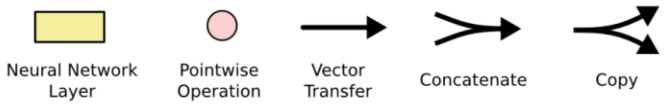


Figure 11 Symbols

Each line carries an entire vector from the output of one node to the inputs of others. The pink circle represents pointwise operations, like vector addition or multiplication while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denotes its content being copied and the copies going to different locations. (Understanding LSTM Networks, 2015)

In this project, an LSTM model is used to predict temperature and failure of a refrigerator. The LSTM model can be used to perform a time series analysis on the data collected periodically. The temperature data collected by the sensor is given as an input to the LSTM model. The LSTM model is a sequential model which include input layer, hidden layer and output layer.

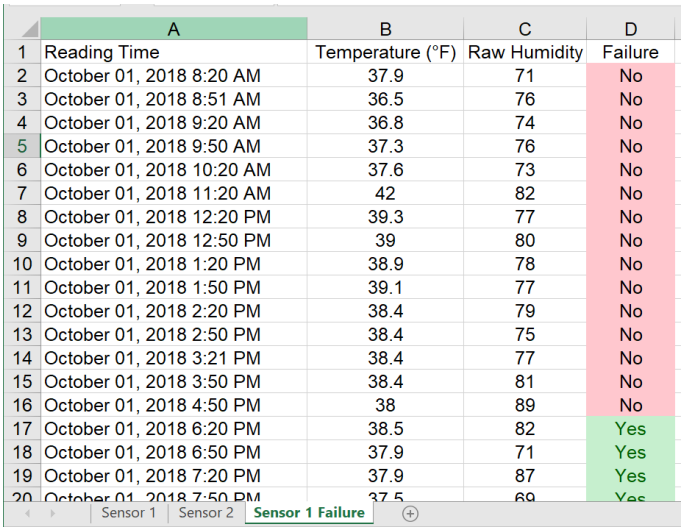
6 Data collection

In this section, I discuss data collection and tools that were used. Verisolutions Inc. has designed a system to collect and monitor temperature and humidity data periodically using sensor modules. Their customers can install multiple sensors in different applications. Each sensor collects temperature and humidity every 30 minutes. The customer can see this data in their user account in **cloud.verisolutions.co**. This project uses raw data from each sensor in the form of an Excel sheet for each sensor.

As shown in Figure 12, every Excel file has 4 columns.

1. Reading time (Date, time)
2. Temperature (F/C)
3. Raw Humidity
4. Failure (1 = Unit failed, 0 = Unit working normal)

This Excel file is used as raw data input for a Python script.



	A	B	C	D
1	Reading Time	Temperature (°F)	Raw Humidity	Failure
2	October 01, 2018 8:20 AM	37.9	71	No
3	October 01, 2018 8:51 AM	36.5	76	No
4	October 01, 2018 9:20 AM	36.8	74	No
5	October 01, 2018 9:50 AM	37.3	76	No
6	October 01, 2018 10:20 AM	37.6	73	No
7	October 01, 2018 11:20 AM	42	82	No
8	October 01, 2018 12:20 PM	39.3	77	No
9	October 01, 2018 12:50 PM	39	80	No
10	October 01, 2018 1:20 PM	38.9	78	No
11	October 01, 2018 1:50 PM	39.1	77	No
12	October 01, 2018 2:20 PM	38.4	79	No
13	October 01, 2018 2:50 PM	38.4	75	No
14	October 01, 2018 3:21 PM	38.4	77	No
15	October 01, 2018 3:50 PM	38.4	81	No
16	October 01, 2018 4:50 PM	38	89	No
17	October 01, 2018 6:20 PM	38.5	82	Yes
18	October 01, 2018 6:50 PM	37.9	71	Yes
19	October 01, 2018 7:20 PM	37.9	87	Yes
20	October 01, 2018 7:50 PM	37.5	69	Yes

Sensor 1 | Sensor 2 | Sensor 1 Failure

Figure 12 Raw data in excel file

6.1 Tools and Software

In this project, the PyCharm IDE was used to develop Python scripts. Python programming language is widely used for machine learning algorithm application. In this project, Python 3.6 platform is used. All scripts are developed using four major packages.

NumPy – used for arrays

Matplotlib – used to create plots

Pandas – used for series and data frames

Keras – used for Machine Learning models

Keras

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. (Keras, n.d.). A model is a core data structure in Keras. A sequential model is the simplest type of Neural Network which creates a linear stack of layers.

1. Steps to create a model:
```
from keras.models import Sequential

Model = Sequential()
```
2. Many layers can be stacked by using the function .add():
```
from keras.layers import Dense

Model.add(Dense(units=64, activation='relu', input_dim=100))

Model.add(Dense(units=10, activation='softmax'))
```
3. Learning process configuration:
```
Model.compile(loss='categorical_crossentropy',

          Optimizer='sgd',

          Metrics=['accuracy'])
```
4. Training data in batches:
```
Model.fit(x_train, y_train, epoch=5, batch_size=32)
```

7 Regression Model Results

In this section, I discuss the application of some of the Machine Learning techniques discussed earlier to the data collected. The goal is the attempt to find the model that can best predict refrigerator performance.

7.1 Polynomial Regression using Numpy

In this study, polynomial regression with various degrees is applied to the 336 (1 week) temperature data points. The results can be compared with the R squared values of input data and polynomial equation data of the Sensor 4. The sensor 4 data show periodical behavior with repeating temperature spikes, so the sensor 4 data was used for prediction and analysis. Three iterations are done using polynomial degree of 20, 35, and 45 (Figure 13-15). The graph shows a comparison of the real values and the polynomial-fit values.

1.

Sensor ID: 4

Total data points: 336

Algorithm: numpy.polyfit

Degree: 20

R squared value: 0.20

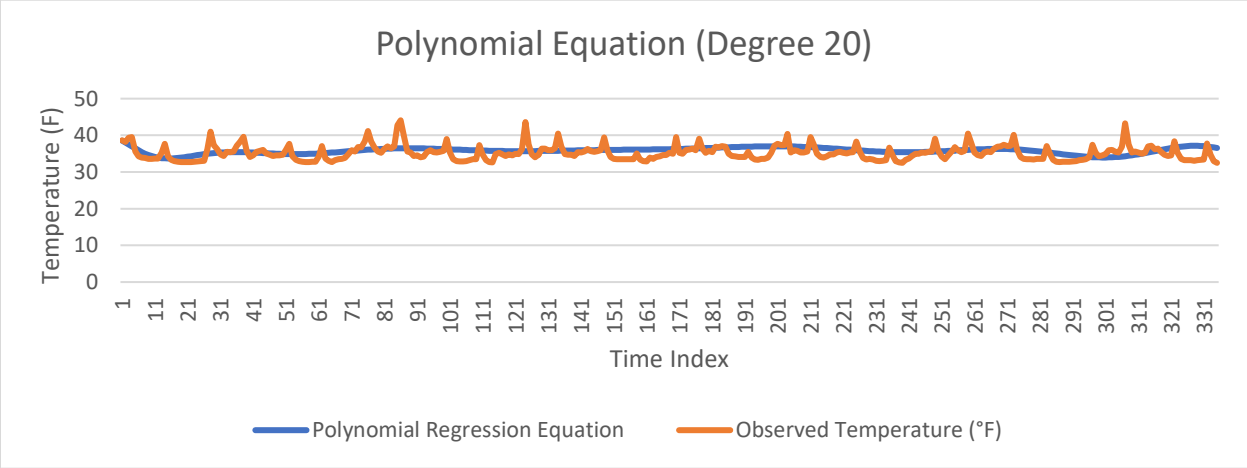


Figure 13 Polynomial Regression (Degree 20)

2.

Sensor ID: 4

Total data points: 336

Algorithm: numpy.polyfit

Degree: 35

R squared value: 0.26

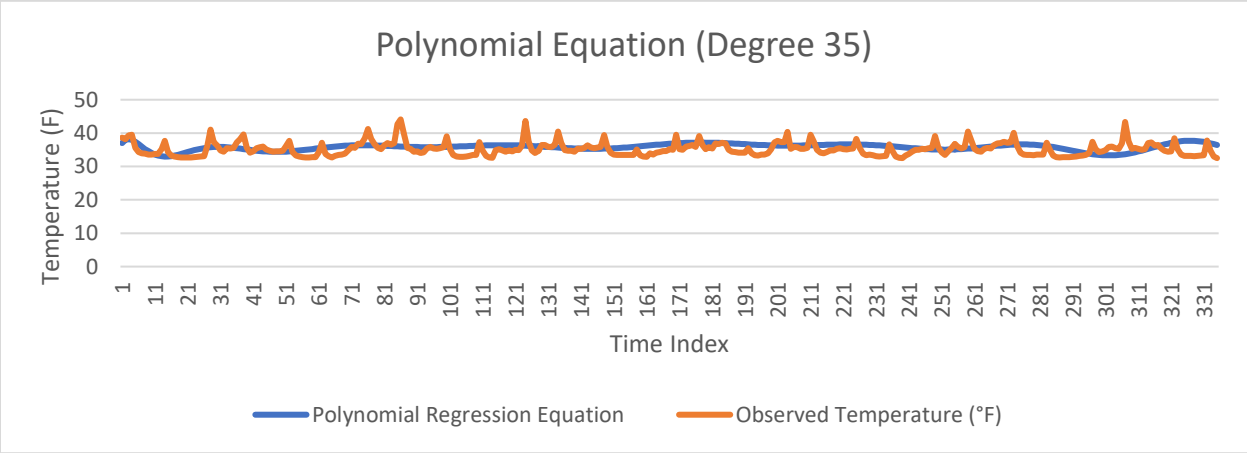


Figure 14 Polynomial Regression (Degree 35)

3.

Sensor ID: 4

Total data points: 336

Algorithm: numpy.polyfit

Degree: 45

R squared value: 0.30

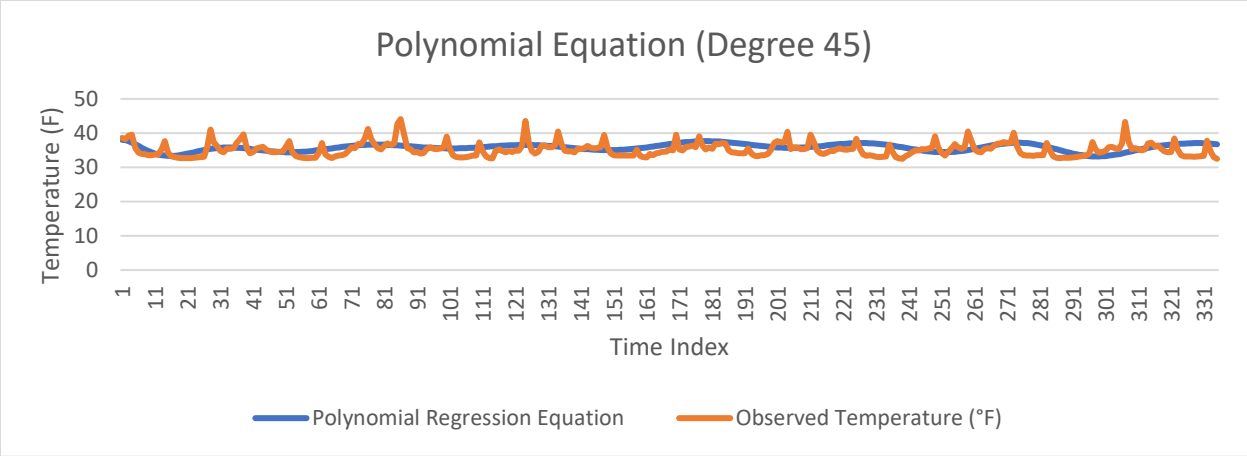


Figure 15 Polynomial Regression (Degree 45)

Analysis

As shown in Figure 13,14, and 15, polynomial regression can find the temperature pattern over time. The R squared value is higher with the higher degrees of polynomial indicating a better fit. However, all the results show that the temperature follows high-low cycle over time. This result shows that polynomial regression is not able to find temperature peaks. It may be possible to find the temperature peaks with a polynomial of degree 336 or higher, but at present, this is computationally intractable. Therefore, polynomial regression won't be pursued further as viable model in this study.

7.2 Support Vector Regression

Support Vector Regression (SVR) model was used to find temperature pattern over 379 data points. The data collection interval for the sensor is 30 minutes. For this test, data of 8 days was used to see the temperature pattern of one week. In addition, the data of one more day was also used to see how the temperature behaves on the same day of the next week.

Sensor ID: 4

Data collection time: October 07, 2018 12:12 AM – October 14, 2018 11:41 PM

Total data points: 379

Algorithm: SVR(kernel='rbf', C=1e3, gamma=0.1)

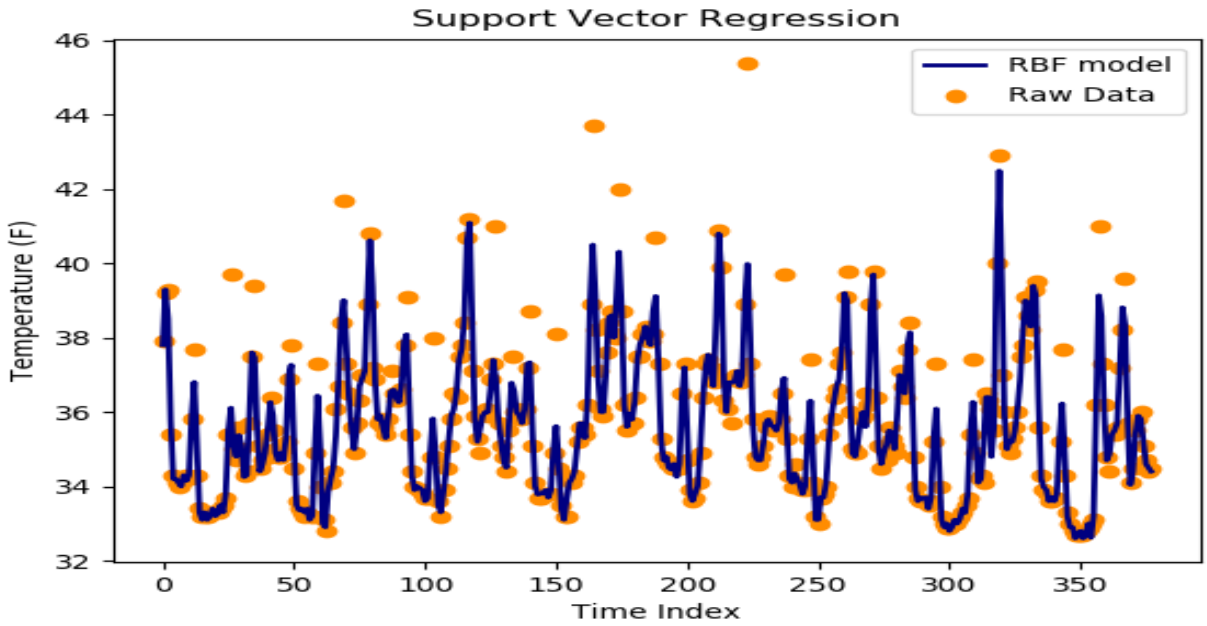


Figure 16 RBF Model vs Raw Data

Analysis

As shown in the Figure 16, the SVR model can capture the high-low temperature data cycle including several of the peak values. This method can be an option for predicting temperature and failure. In this paper, the SVR model is not used for more analysis to focus on finding other methods for prediction. It is mentioned in the future work to be studied further.

8 LSTM Model Results

The LSTM model requires parameter tuning of neuron count, input window size, output size, batch size, and epoch to fit the model. Before applying the LSTM model to the data, I experimented with the input window size and neuron count to see what the best options would be.

8.1 Root Mean Square Error (RMSE) Comparison

In this study, RMSE comparison is given between predicted and real temperature values. The length of input data (number of temperature samples) is variable to see how much input data would provide a better prediction with lower RMSE value. For this study, Sensor 3 is used because it has periodical temperature cycles, which will give better understanding of the temperature profile and its prediction.

Sensor – Sensor 3

window_size = Variable input length (independent variable)

output_size = 48 (1 day) # output window, number of predictions

neuron_count = 256 # number of neuron to be used for training

batch_count = 50 # batch size

epoch_count = 100 # number of epoch

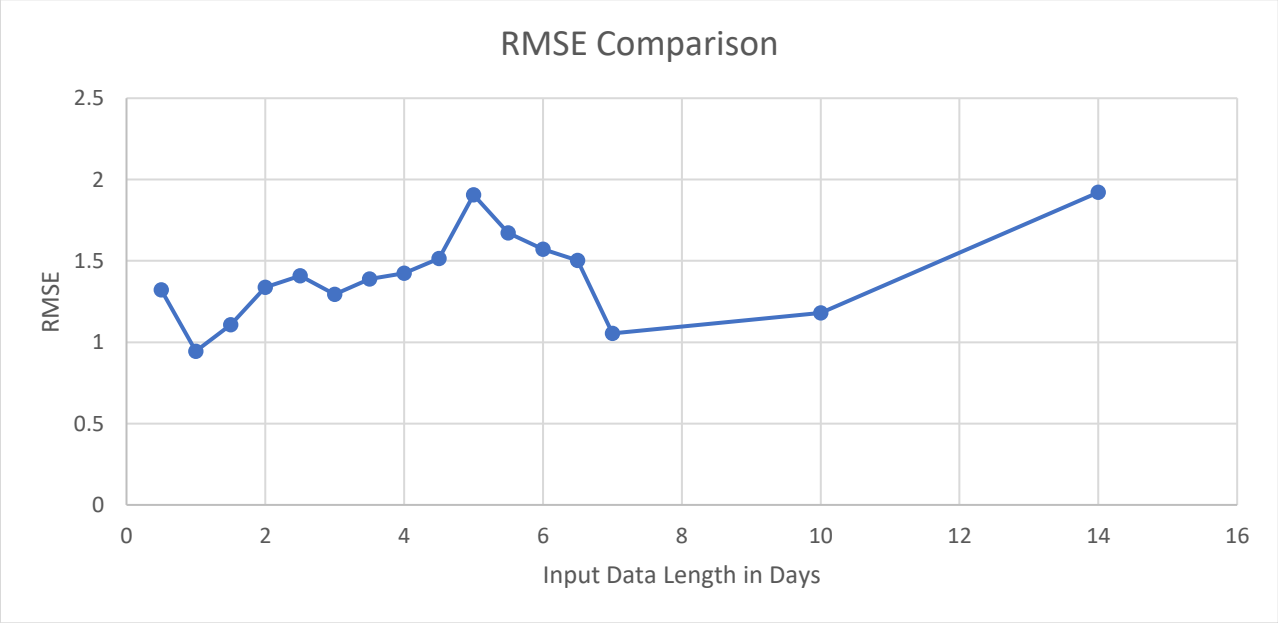


Figure 17 RMSE Comparison for different Input Data Length in Days

Analysis

As shown in Figure 17, the RMSE value is different for each length of input data. However, the RMSE value is lowest with the input length of 1 day and 1 week. The RMSE error is higher with length of above 1 week. The restaurant kitchen operation follows a daily schedule based on the day of the week. Based on this result, an input window of 1 week is used for the rest of the study. The input window of 1 day is mentioned in Future Work.

8.2 Neuron count comparison

In this study, temperature prediction is compared by training the neural network with different neuron counts (Figure 18). The input data is 336 temperature data points, which is one week of data. The output is 48 data points, which is one day. The training and test are done on the sensor 3 data.

Sensor ID: Sensor 3

window_size = 336 # input window, number of input values

output_size = 48 # output window, number of predictions

batch_count = 50 # batch size

epoch_count = 100 # number of epoch

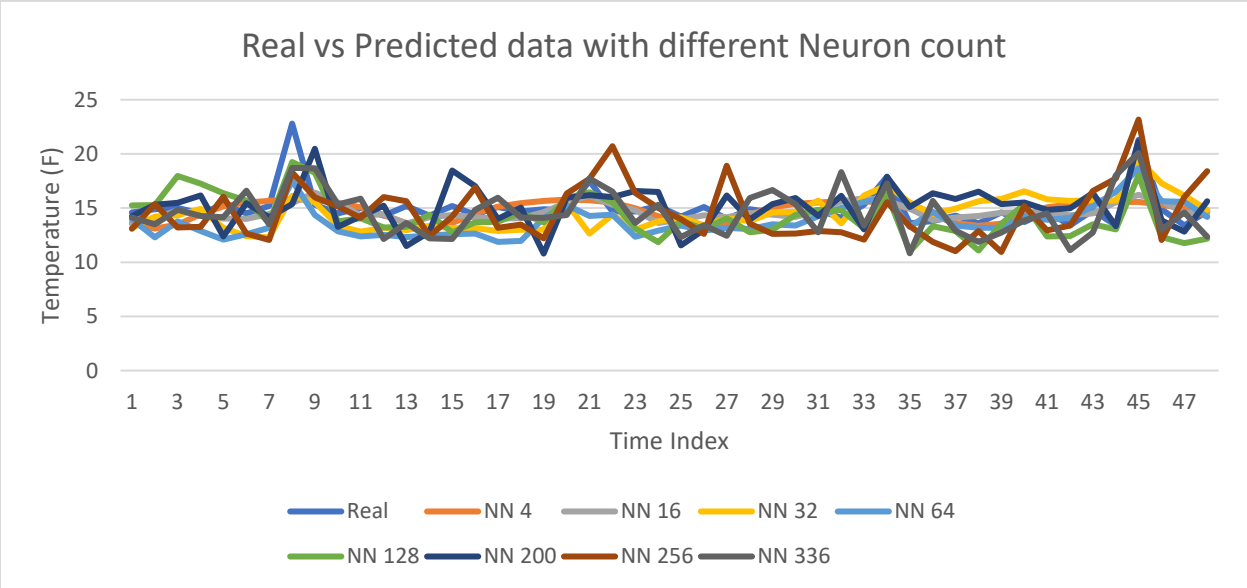


Figure 18 Real vs Prediction at different Neuron count

RMSE values for figure 18:

Neuron Count	RMSE
4	1.67
16	1.55
32	1.93
64	1.92
128	1.76
200	2.14
256	2.32
336	1.88

Table 1 RMSE value for figure 17

As shown in the Table 1, the RMSE value is lowest for a neuron count of 16. It would be considered a best fit, but it cannot capture the peak value, which is necessary to predict the cooler failure. The value of 4 underfits the data as shown in Figure 19 whereas a value of 256 overfits the data as shown in Figure 20.

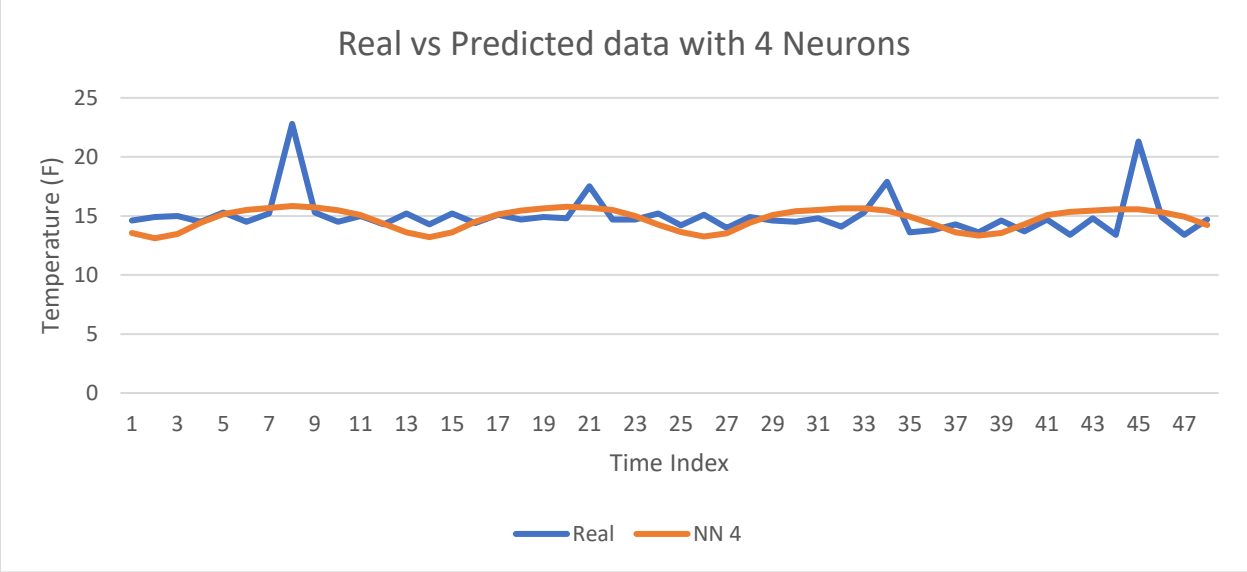


Figure 19 Underfitting

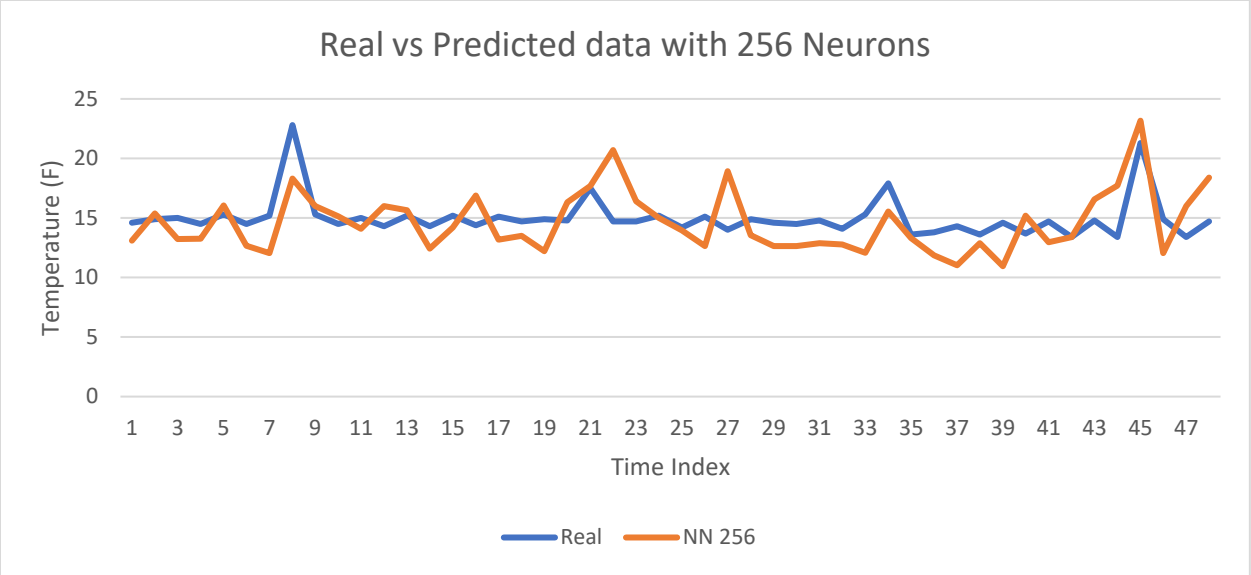


Figure 20 Overfitting

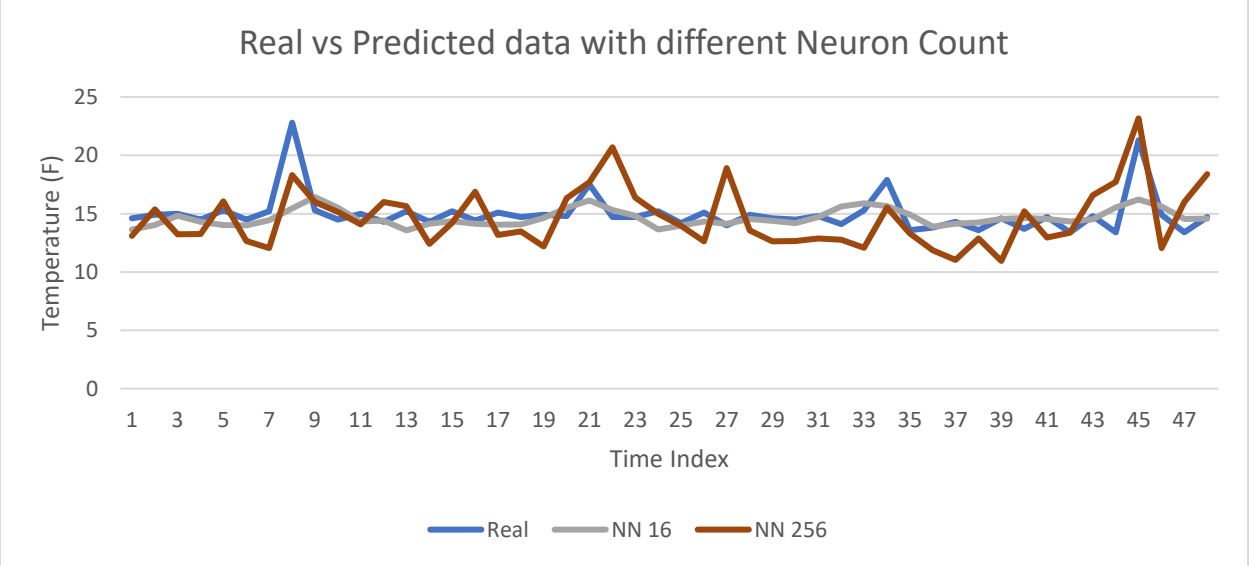


Figure 21 Best Fit (NN 16)

Analysis:

As shown in Figure 18, the predicted temperature is different for each neuron count. By increasing the neuron count, the LSTM model can better predict the temperature peaks of real data. The RMSE values alone do not fully reflect this observation because the error is higher on the non-peak temperature values. Though a neuron count of 16 reduces the RMSE, it cannot predict temperature peaks which is more

important to predict failure. Overall, a neuron count of 256 appears to perform better than other values. With a neuron count of 336, the model can find peaks with lower RMSE values with the cost of higher computation time. As a result, the value of 256 is chosen for the rest of the study. The neuron count value of 336 is mentioned in the Future Work.

8.3 Training and Testing with the model of another sensor

In this section, I investigate whether the data from one sensor can be used to predict the data of another sensor. In this study, data from sensor 3 is used for testing the neural network, after it has been trained using sensors 1,2,3,4 and 5. Sensor 3 has one instance of failure with the temperature being out of range and one instance of out of range temperature without failure. It also has a periodic up-down cycle, which makes the data of sensor 3 suitable for training the model. The input data is temperature data points and the output data is the probability of failure. After training the neural network with the data from sensor 3, the RNN was tested using the data of sensor 3. The temperature is the input and probability of failure is the output.

The training parameters are as follow:

Neuron Count – 4

Input window size – 1

Output size – 1

Batch size – 100

Epoch – 100

1. Train Sensor 3 – Test Sensor 3 (Figure 22)

   Training Data Length – 7685 (160 days)

   Testing Data Length – 7685 (160 days)

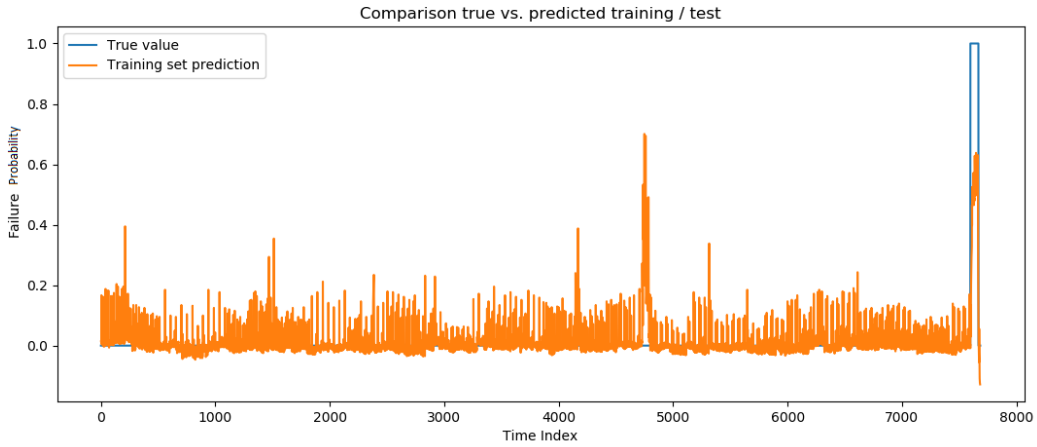


Figure 22 Train Sensor 3 - Test Sensor 3

2. Train Sensor 1 – Test Sensor 3 (Figure 23)

   Training Data Length – 16142 (336 days)

   Testing Data Length – 7658 (160 days)

Figure 23 Train Sensor 1 - Test Sensor 3

3. Train Sensor 2 – Test Sensor 3 (Figure 24)

   Training Data Length – 12680 (264 days)

   Testing Data Length – 7658 (160 days)

Figure 24 Train Sensor 2 - Test Sensor 3

4. Train Sensor 4 – Test Sensor 3 (Figure 25)

   Training Data Length – 7686 (160 days)

   Testing Data Length – 7658 (160 days)

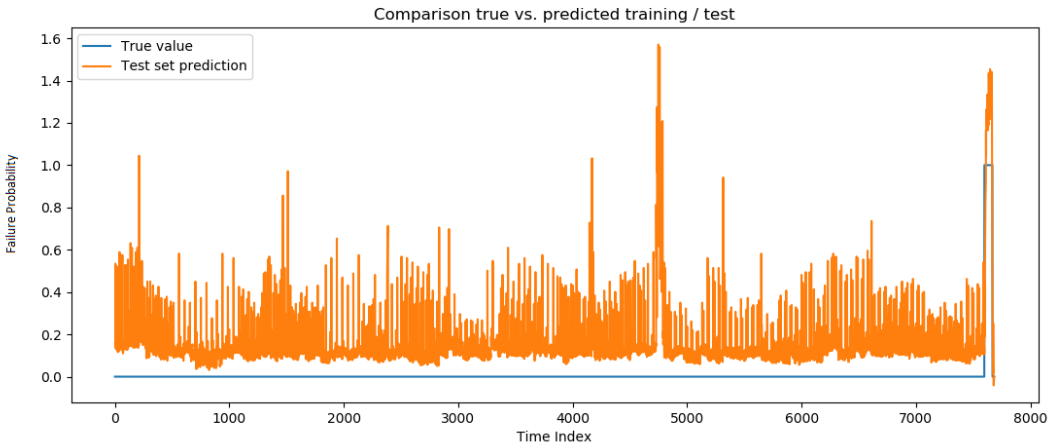


Figure 25 Train Sensor 4 - Test Sensor 3

5. Train Sensor 5 – Test Sensor 3 (Figure 26)

   Training Data Length – 6767 (140 days)

   Testing Data Length – 7658 (160 days)

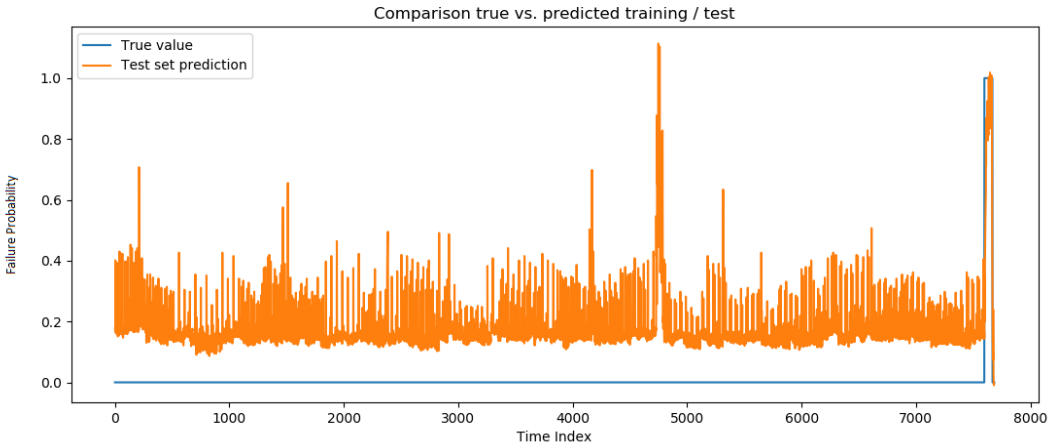


Figure 26 Train Sensor 5 - Test Sensor 3

Analysis

Each refrigerator has a different threshold of failure as well as temperature profiles, which means that the min, max and average temperatures of each sensor are different. All sensors have different temperatures considered as failure. For example, 50 F may be considered failure for unit 1, whereas 75 F may be considered failure for unit 2. This study shows that, the data of one sensor cannot be used to predict the failure of another sensor. If the sensors are installed in the same type of coolers, and if they have same temperature profile, then one sensor data might be used to test another sensor.

8.4 Temperature Prediction – Sensor Groups

In this study, the sensors are selected based on group. A group of sensors are ones installed in similar coolers used for similar purposes with similar profiles (Table 2). The group of sensors used in this study are for *sea-food* coolers. For this test, the data of one sensor with cooler failure is used to train the model. The trained model is used to test four other sensors of the same group (Figure 28-31).

Temperature profile

Sensor	Min	Max	Average
Sea-food 1	**32.1**	**53.7**	**37.6**
Sea-food 2	**24.9**	**65.1**	**36.7**
Sea-food 3	**32.6**	**52.6**	**37.82**
Sea-food 4	**28.3**	**61.2**	**36.9**
Sea-food 5	**32.0**	**56.8**	**36.2**

Table 2 SeaFood Cooler Group Temperature Profile

Train sensor – Sea-food sensor 1 (Figure 27)

Input Data – 336 temperature data points (1 week)

Output Data – 48 temperature data points (1 day)

Parameters tuning

window_size = 336 # input window, number of input values
output_size = 48 # output window, number of predictions
neuron_count = 256 # number of neurons to be used for training
batch_count = 50 # batch size
epoch_count = 100 # number of epoch

Training Data

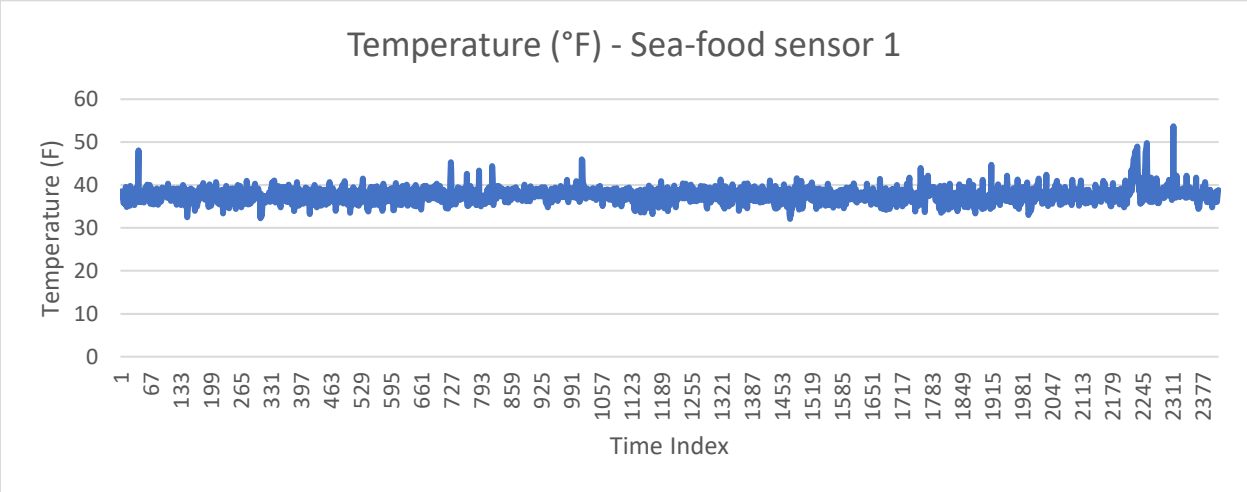


Figure 27 SeaFood Sensor 1 Temperature

Sensor 2

RMSE score – 2.74

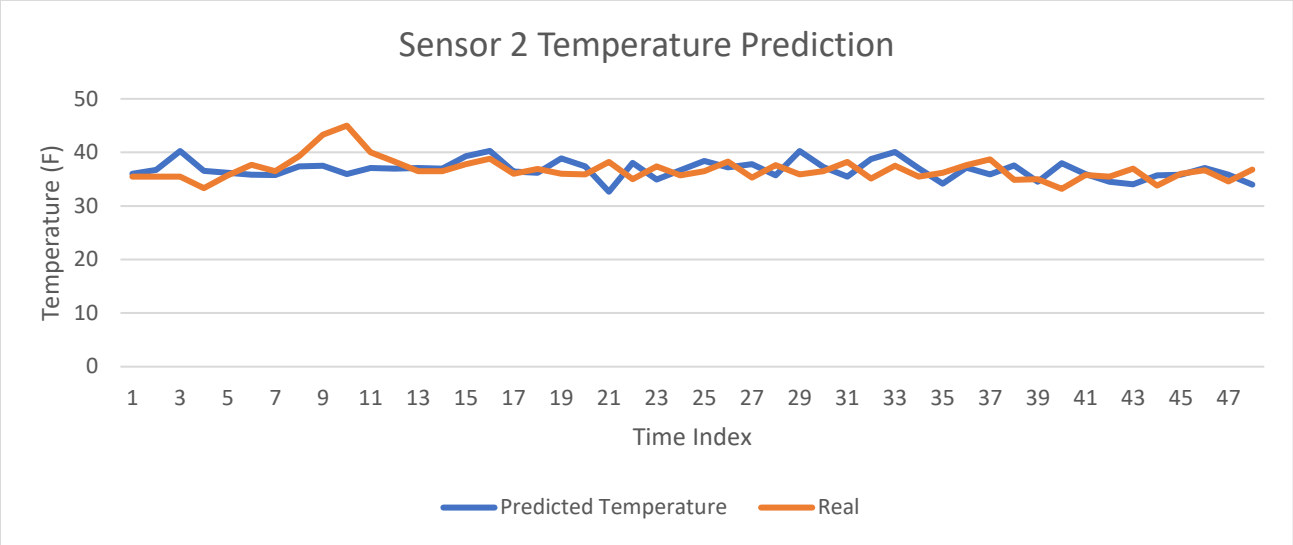


Figure 28 SeaFood Sensor 2 Temperature Prediction

Sensor 3

RMSE score – 2.05

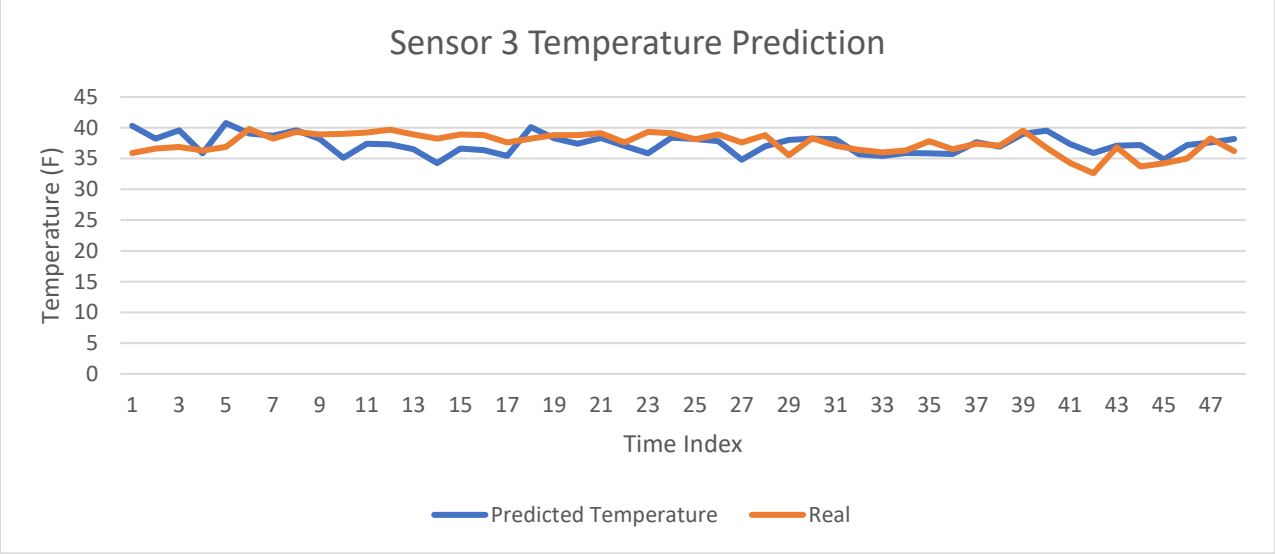


Figure 29 SeaFood Sensor 3 Temperature Prediction

Sensor 4

RMSE score - 5.03

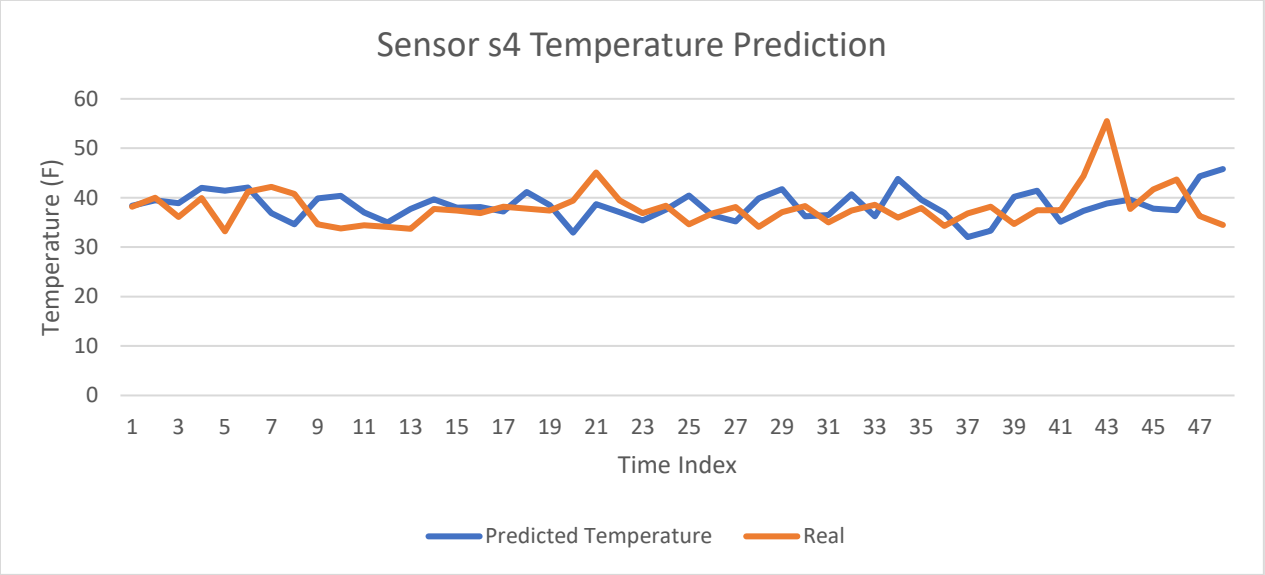


Figure 30 SeaFood Sensor 4 Temperature Prediction

Sensor 5

RMSE score – 3.67

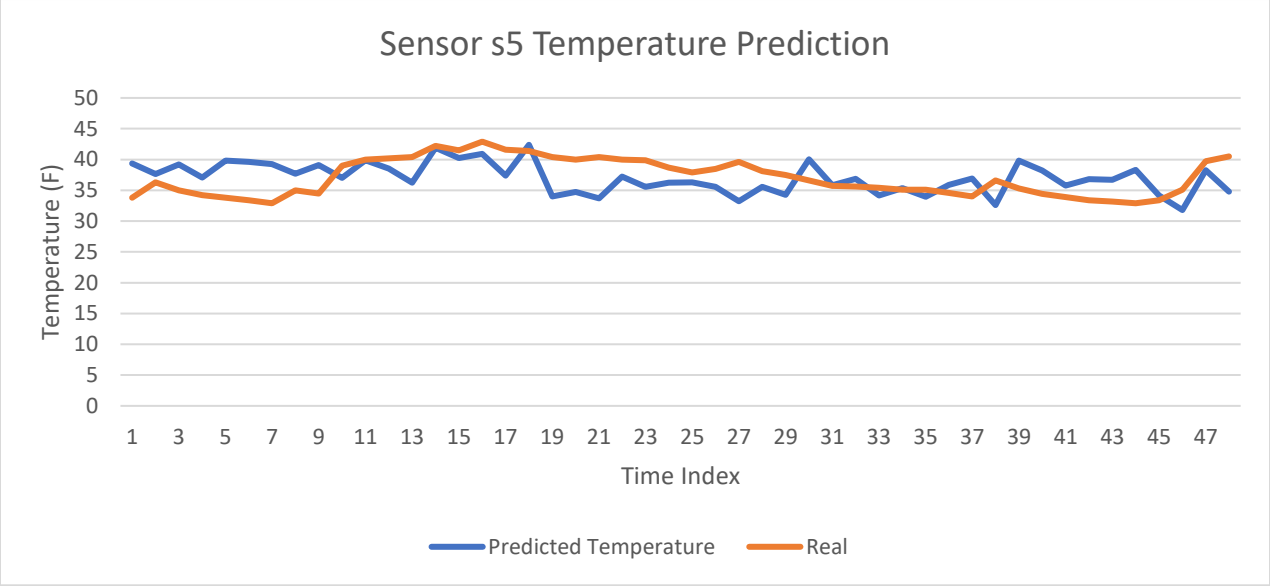


Figure 31 SeaFood Sensor 5 Temperature Predictions

Analysis

As shown in Table 2, the min, max and average temperature of all sea-food coolers are similar. So, a model trained with the data of one sensor in the group can be used to predict the other sensor of the same

group. However, I did not continue investigating the idea of group-wise prediction, because, although sea-food coolers all seem to have very similar profiles, this is not the case with other groups of coolers such as Walk-In coolers. Group-wise prediction is left as future work. Instead, investigated whether temperature data from one cooler could be used to predict the future temperature of the same unit, which is presented in the next sub-section.

8.5 Two Model Approach

In this section, I will investigate whether Machine Learning can adequately predict refrigerator performance based upon temperature data gathered from the same unit. The ultimate goal is to predict refrigerator failure before it occurs to give users adequate time to address the failure, perhaps even to avoid it.

Based upon the results from previous sections, I have decided to use an LSTM with a Neuron Count of 256 to model the temperature data. I have also determined that it is best to use a cooler's own sensor data to predict its future data instead of using one cooler's data to predict another.

In order to predict the temperature of a refrigerator using its own historical data and to use an input size of one week of data, I will be using a sliding window. The sliding window will be 336 data points, which will be used to predict the next 48 data points (1 day). This model will also be useful in practice, where each day, new data is collected, adding to the data of the last 6 day, then the next day's temperature is predicted. I will also separate the modelling of the temperature and the modelling of cooler failure.

In this section, I investigated about how a model can be divided into two parts to predict the temperature and the failure. The first part is to predict the temperature of 1 day using 7 days' worth of data as input. The second part is to find the failure in the predicted temperature of part one. The input data for model 1 and model 2 is stored in an Excel file. It contains three columns titled Time, Temperature, and Failure.

Steps to predict failure:

1. Import an Excel file with historical data of temperature and failure separated by 30 minutes time in sequence
2. Prepare input data as required by Keras functions

3. Train Model 1 with whole data except last week using sliding window of input size of 1 week and output size of 1 day of data to predict temperature
4. Test the model by giving last week of data as input to trained model
5. Store data of the result of Model 1
6. Train Model 2 with whole data with temperature as input and failure as output
7. Test the Model 2 with the output data of Model 1 as input to find failure probability

The following example demonstrates Model 1 and Model 2 using fictitious data of 19 samples to predict 4 outputs (Table 3). The following table is an example representation of the model and the data reshape. All the tests are performed on the real data collected from a cooler.

1. Input data for learning model

	Time	Temperature	Failure
Training	12:00	1	0
	12:30	2	0
	13:00	3	0
	13:30	4	0
	14:00	5	0
	14:30	6	0
	15:00	7	0
	15:30	8	0
	16:00	9	0
	16:30	10	0
	17:00	11	0
	17:30	12	1
	18:00	13	1
	18:30	14	1
	19:00	15	1
	19:30	16	0
	20:00	17	0
	20:30	18	0
	21:00	19	0
Test	21:30	X	X
	22:00	X	X
	22:30	X	X
	23:00	X	X

Table 3 Raw data split

2. Data reshaping for input to LSTM model

   The input data is stored in an Excel document. The Keras library only accepts 3D data in sequential format. The reshaping process converts the single array input data into 3D data with the sliding window of 1 week. The Table 4 explains an example representation of sliding window of 5 data points. After reshaping the input data, model 1 is given 5 samples as input to generate 4 output samples. The training is done by sliding the window along the sequence of input and output by 1 sample.

	Input					Output			
Training	1	2	3	4	5	6	7	8	9
	2	3	4	5	6	7	8	9	10
	3	4	5	6	7	8	9	10	11
	4	5	6	7	8	9	10	11	12
	5	6	7	8	9	10	11	12	13
	6	7	8	9	10	11	12	13	14
Test	15	16	17	18	19	Prediction (4 data points)			

Table 4 Temperature Prediction

3. Find failure in the predicted values of last step

   The model 2 is trained to find a relationship between temperature and failure. The predicted data in model 1 (4 samples) is used as an input for model 2 to find failure.

Test	21:30	Predicted 4 Data Points	Failure prediction
	22:00		
	22:30		
	23:00		

Table 5 Failure prediction

8.6 Temperature and Failure prediction before and after failure

This test includes two separate models for temperature prediction and failure prediction as explained in above section. In this study, 2 different tests are performed. The first tests use the data produced before the failure happens. The second test uses 4 hours shifted data after the failure. Each model is trained with the data of 1 week using sliding window. The model 1 is used for predicting temperature of one day in the future using one week of past data. The predicted data from model 1 goes into model 2 as input for

predicting failure. The first test is done by giving input data before failure happens. In the second model, input data is delayed by 4 hours after failure occurrence. For example, if the failure happened at 3 pm of any particular day, the first model will use the data before 3 pm to predict the failure. The second model will use data before 7 pm of that day to predict the failure. For this study, real data collected from six different sensors are used for analysis. Each sensor results includes 5 graphs (Figure 32-61) as explained below.

Graph 1 – Raw sensor data (Temperature and Failure) (Figure 32, Figure 37, Figure 42, Figure 47, Figure 52, Figure 57)

Graph 2 – Temperature prediction before actual failure happens (Input data is last week of data before failure) (Figure 33, Figure 38, Figure 43, Figure 48, Figure 53, Figure 58)

Graph 3 – Failure prediction (Input data is obtained in the Graph 2) (Figure 34, Figure 39, Figure 44, Figure 49, Figure 54, Figure 59)

Graph 4 - Temperature prediction 4 hours after actual failure happens (Figure 35, Figure 40, Figure 45, Figure 50, Figure 55, Figure 60)

Graph 5 – Failure prediction (Input data is obtained in the Graph 4) (Figure 36, Figure 41, Figure 46, Figure 51, Figure 56, Figure 61)

The model training parameters are as follow:

window_size : 336 (1 week of data)

output_size : 48 (1 day of data)

neuron_count : 256

batch_count : 50

epoch_count : 100

8.6.1 Sensor 1 Results (Figure 32-36)

The raw temperature and failure data are given in the Figure 32. As shown in the figure, failure occurred at the data point 14375.

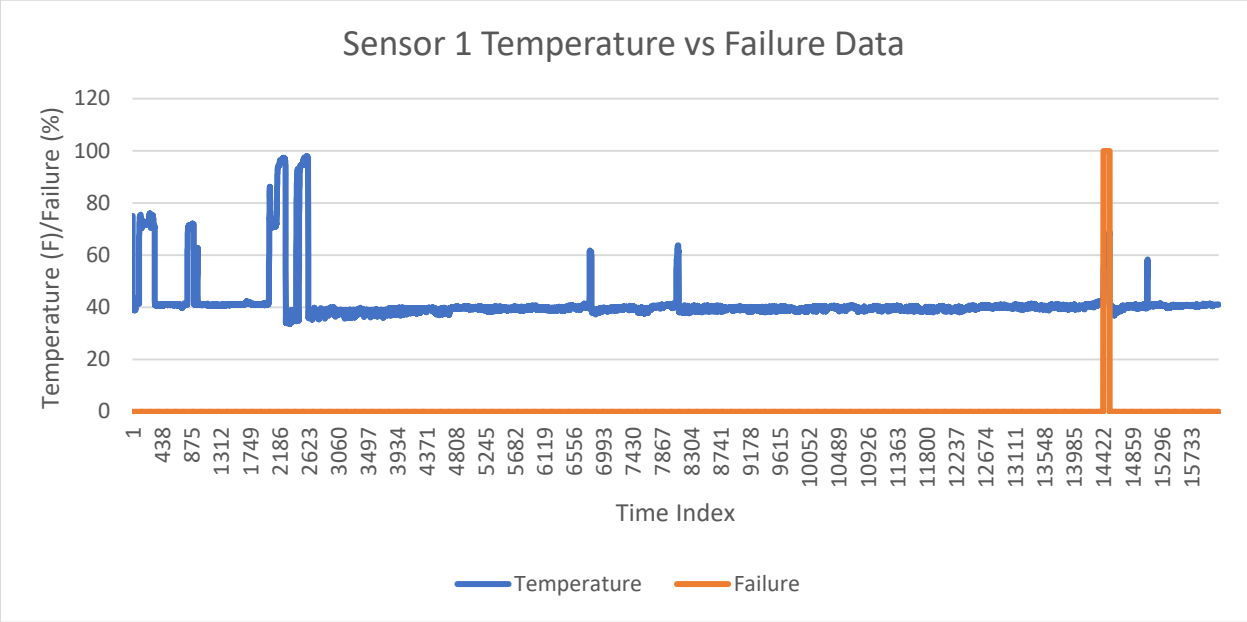


Figure 32 Sensor 1 Temperature vs Failure Data

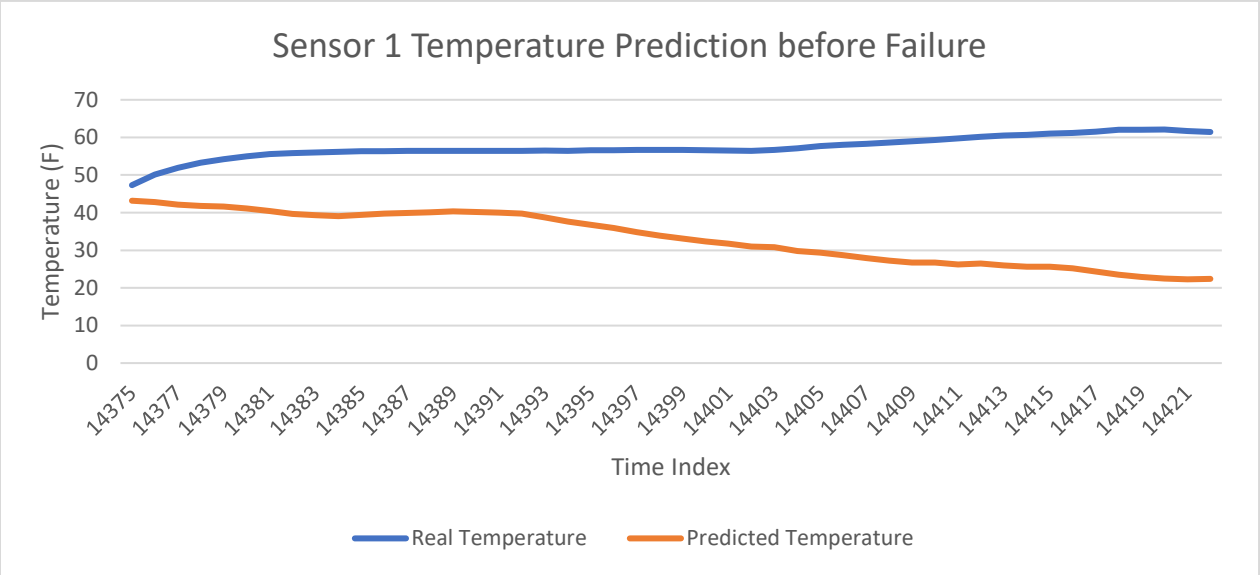


Figure 33 Sensor 1 Temperature Prediction before failure

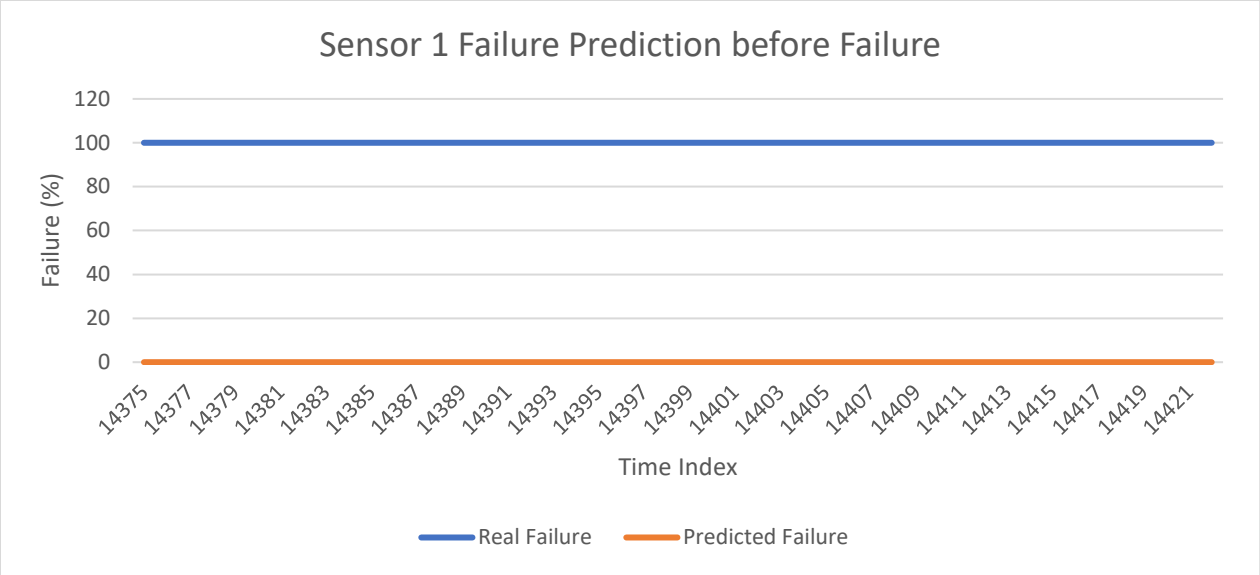


Figure 34 Sensor 1 Failure Prediction before failure

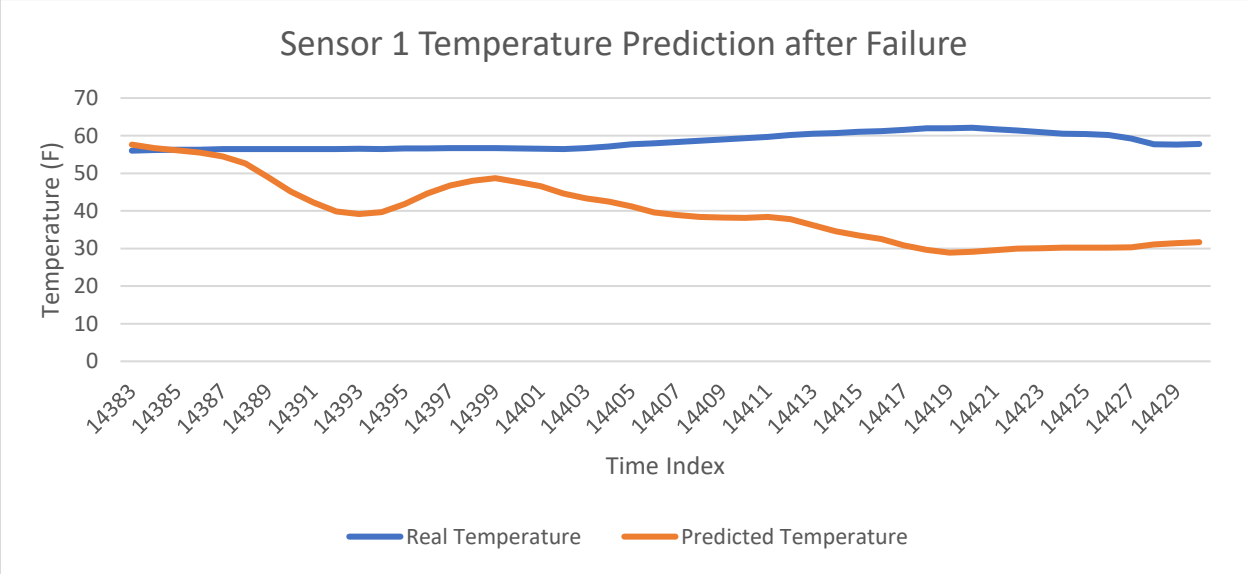


Figure 35 Sensor 1 Temperature Prediction after failure

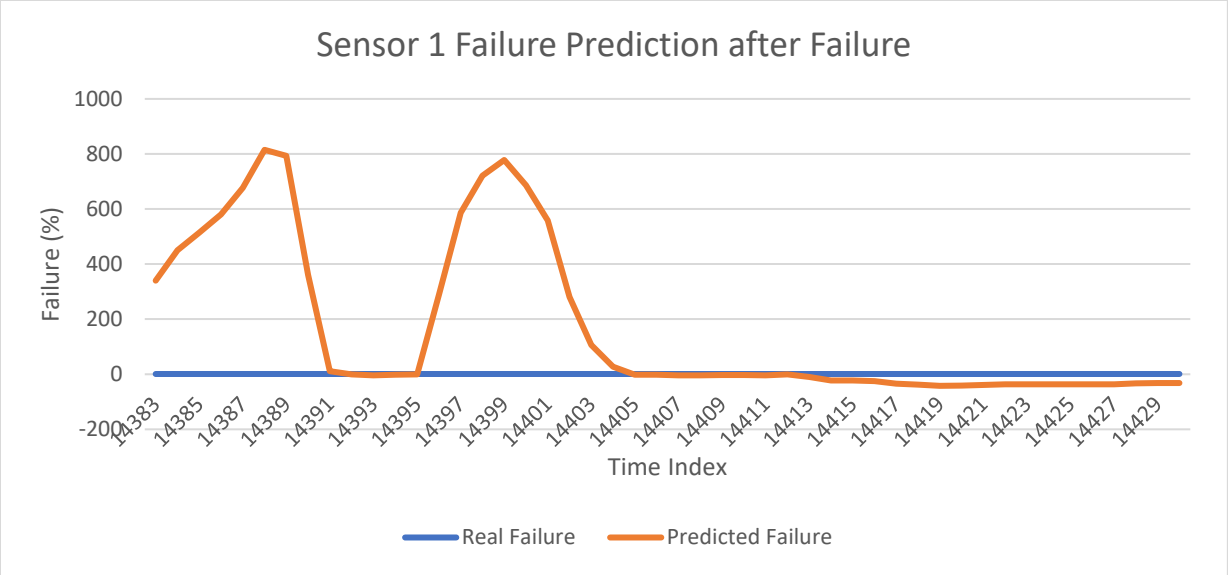


Figure 36 Sensor 1 Failure Prediction after failure

Analysis

The temperature and failure percentage data are given in the Figure 32. The sensor fails at the time index of 14375. As shown in Figure 33 & 34, the model cannot predict the temperature of 1 day in future when the failure happened. As a result, it cannot predict the failure by using the data of past week. For this unit, failure happens suddenly without giving any information in the data of the previous week. As shown in the Figure 35 & 36, the input data was shifted by 4 hours (8 data points) to predict the temperature. However, the model given the *4 hour shifted* input data after failure is able to predict failure partially.

8.6.2 Sensor 2 Results (Figure 37-41)

The raw temperature and failure data are given in the Figure 37. As shown in the figure, failure occurred at the data point 12536.

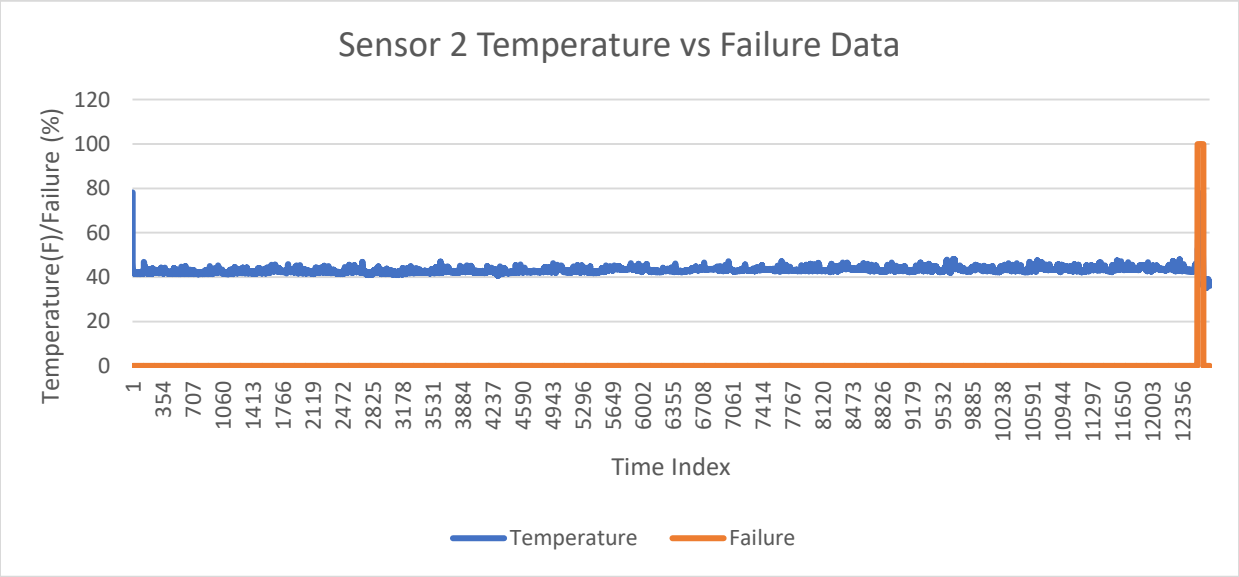


Figure 37 Sensor 2 Temperature vs Failure Data

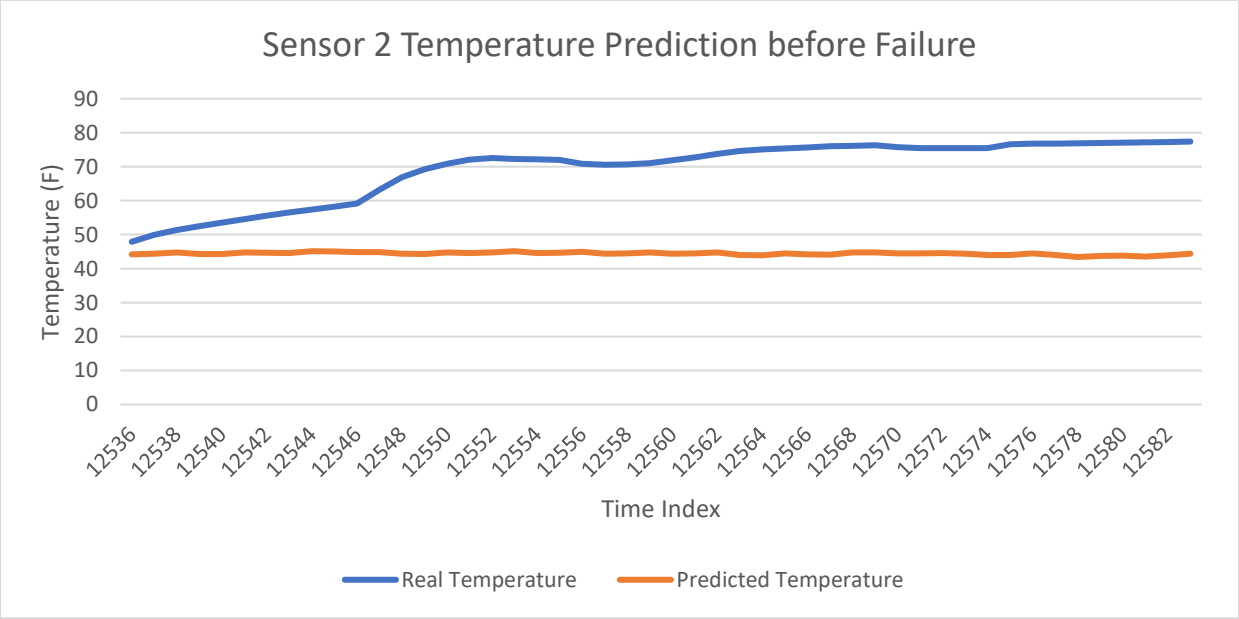


Figure 38 Sensor 2 Temperature Prediction before failure

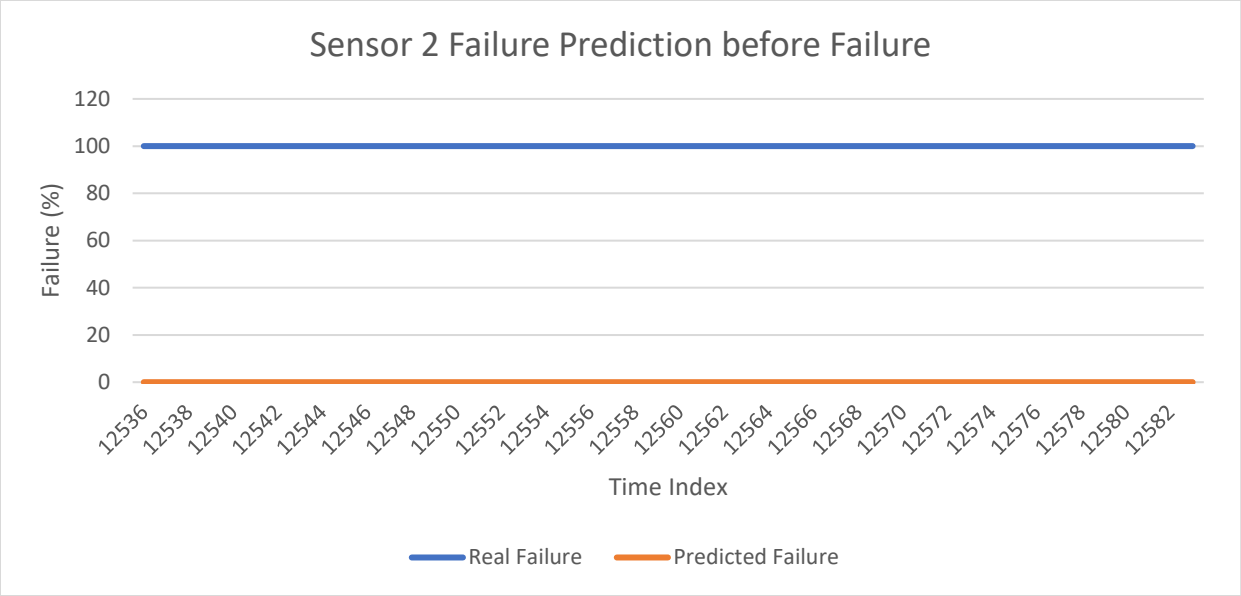


Figure 39 Sensor 2 Failure Prediction before failure

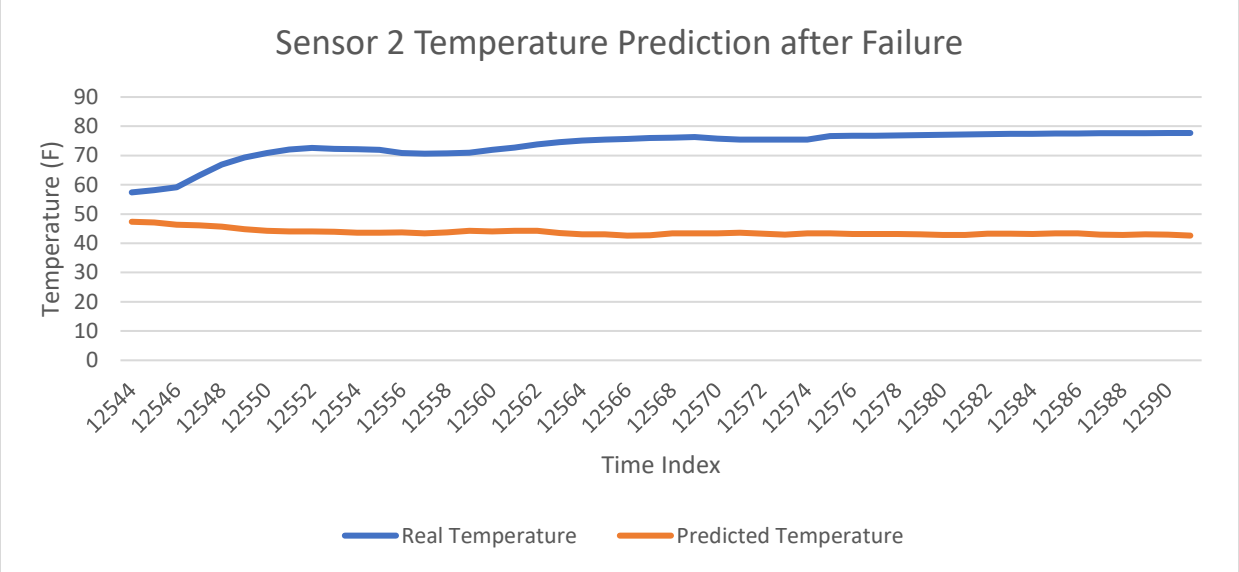


Figure 40 Sensor 2 Temperature Prediction after failure

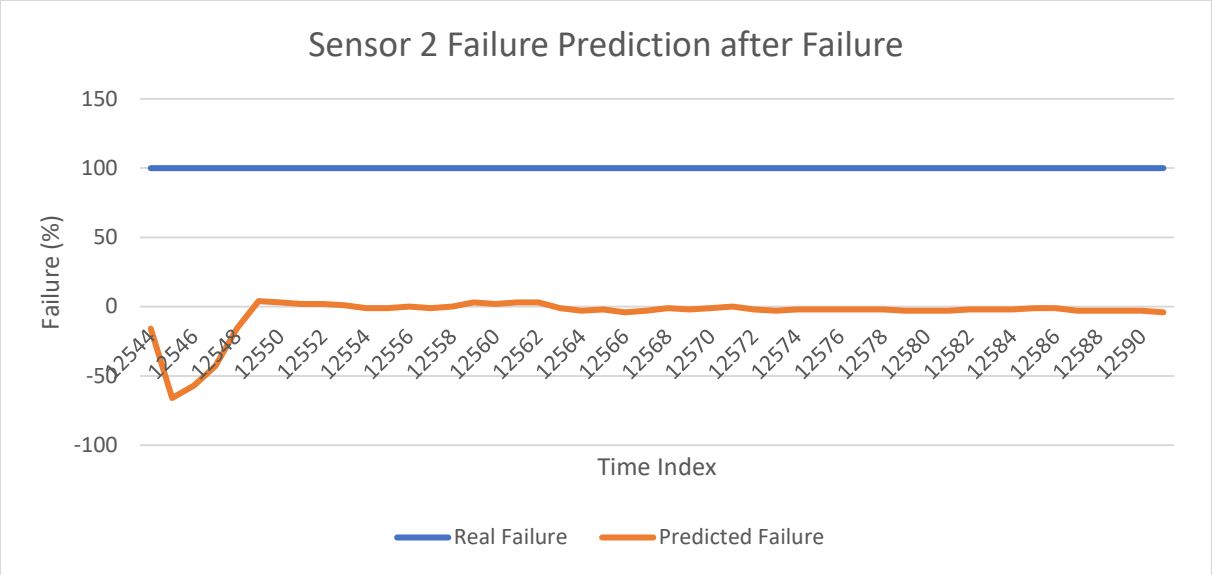


Figure 41 Sensor 2 Failure Prediction after failure

Analysis

The temperature and failure percentage data are given in the Figure 37. The sensor fails at the time index of 12536. As shown in Figure 38 & 39, the model cannot predict the temperature of 1 day in future when the failure happened. As a result, it cannot predict the failure by using the data of past week. For this unit, failure happens suddenly without giving any information in the data of the previous week. As shown in the Figure 40 & 41, the input data was shifted by 4 hours (8 data points) to predict the temperature. The model given the *4 hour shifted* input data is not able to predict failure.

8.6.3 Sensor 3 Results (Figure 42-46)

The raw temperature and failure data are given in the Figure 42. As shown in the figure, failure occurred at the data point 7603.

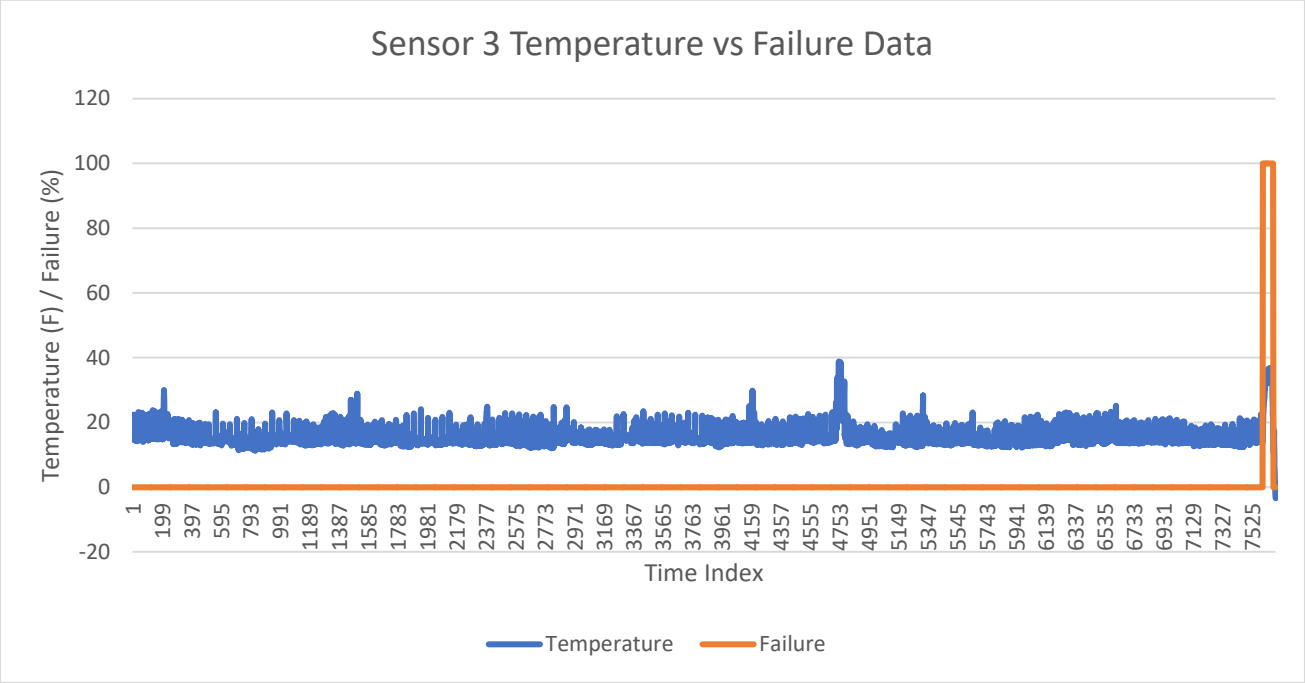


Figure 42 Sensor 3 Temperature vs Failure

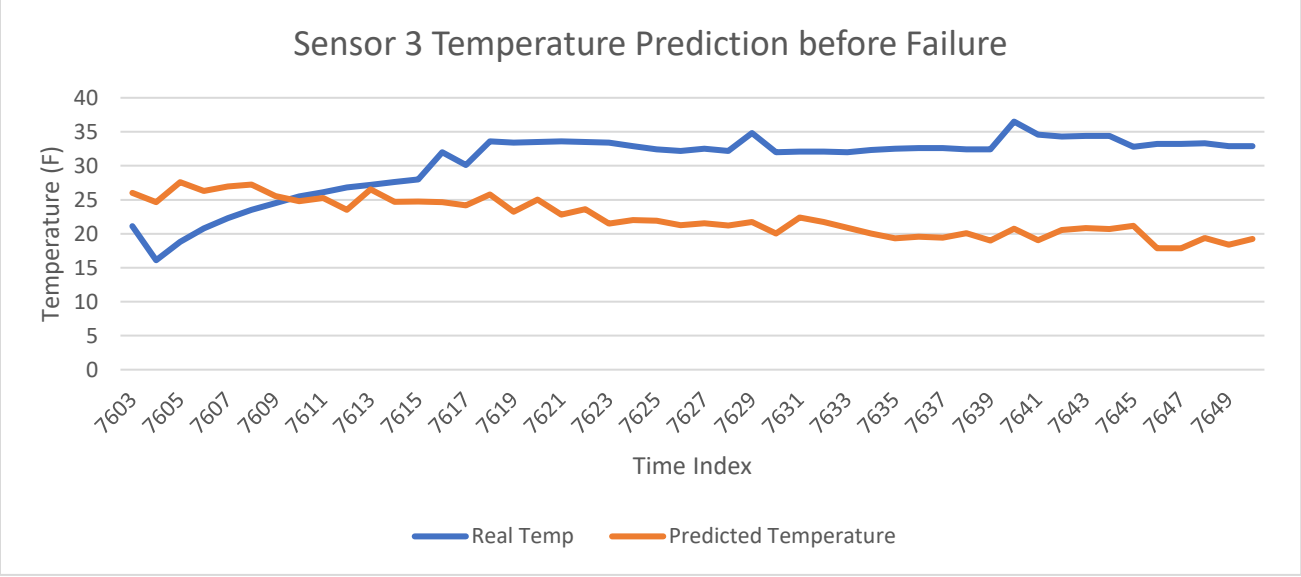


Figure 43 Sensor 3 Temperature prediction before failure

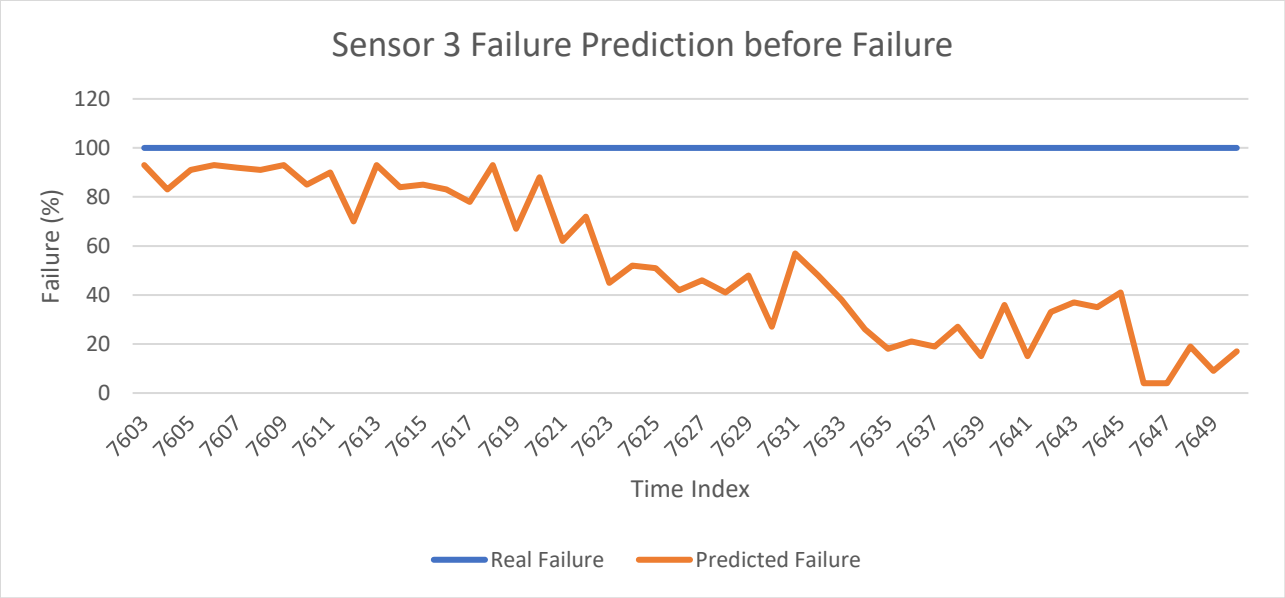


Figure 44 Sensor 3 Failure prediction before failure

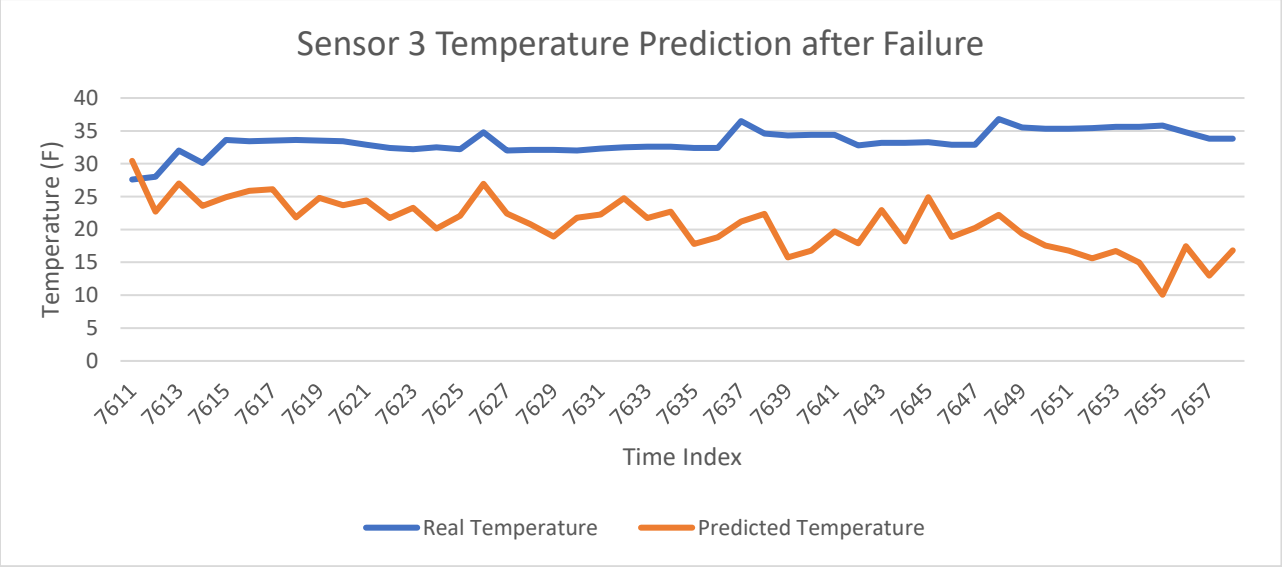


Figure 45 Sensor 3 Temperature prediction after failure

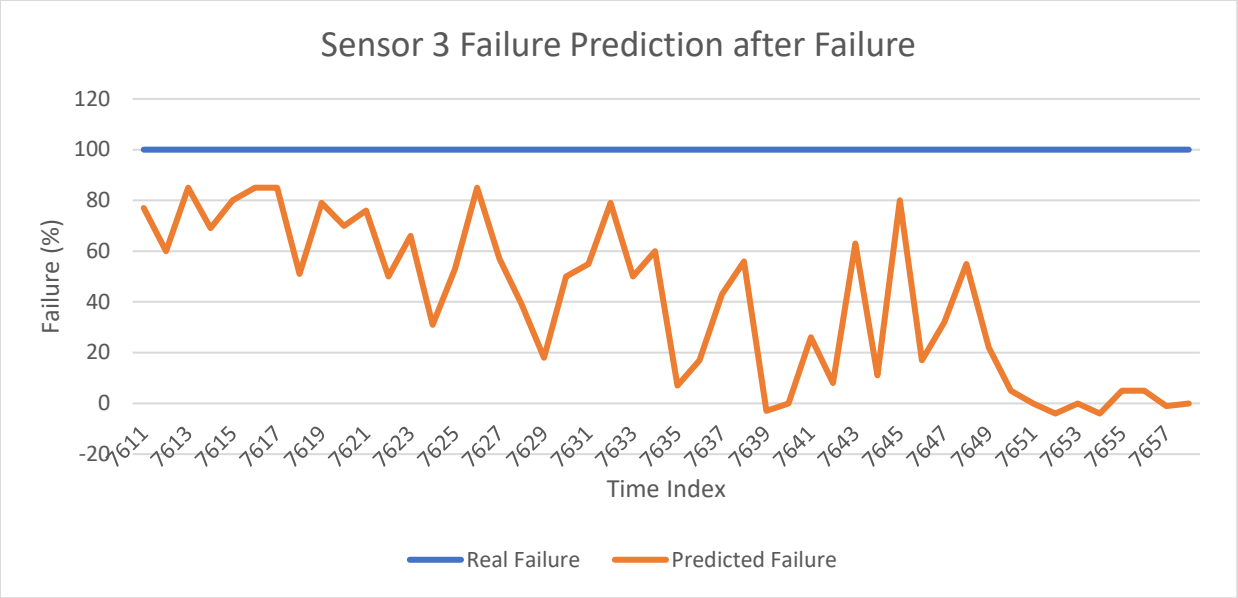


Figure 46 Sensor 3 Failure prediction after failure

Analysis

The temperature and failure percentage data are given in the Figure 42. The sensor fails at the time index of 7603. As shown in Figure 43 & 44, the model can predict the temperature of 1 day in future when the failure happened. As a result, it predicted the failure with above 80% confidence by using the data of past week. For this unit, the temperature is rising at slower pace before it reaches threshold of failure. As shown in the Figure 45 & 46, the input data was shifted by 4 hours (8 data points) to predict the temperature. The model given the *4 hour shifted* input data after failure is able to predict failure with higher confidence score.

8.6.4 Sensor 4 Results (Figure 47-51)

The raw temperature and failure data are given in the Figure 47. As shown in the figure, failure occurred at the data point 11376.

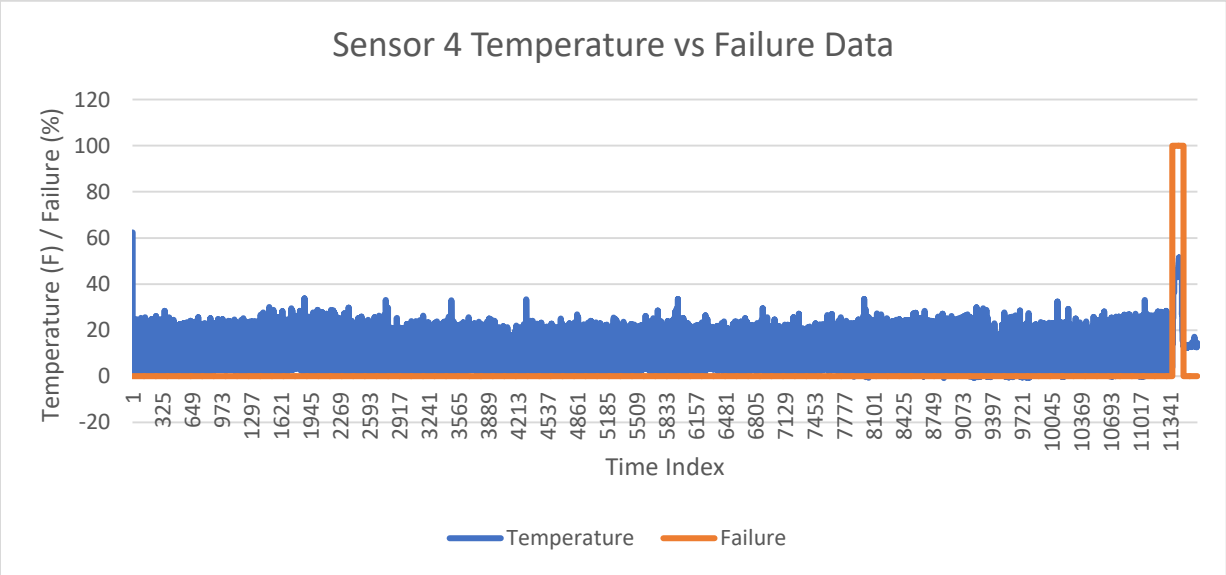


Figure 47 Sensor 4 Temperature vs Failure

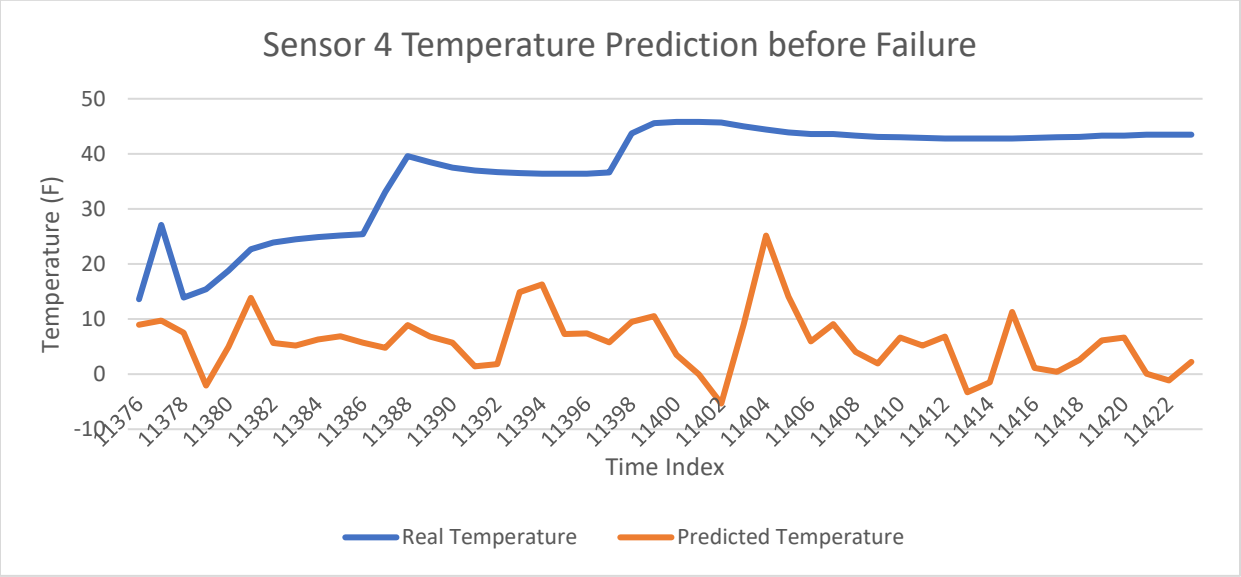


Figure 48 Sensor 4 Temperature prediction before failure

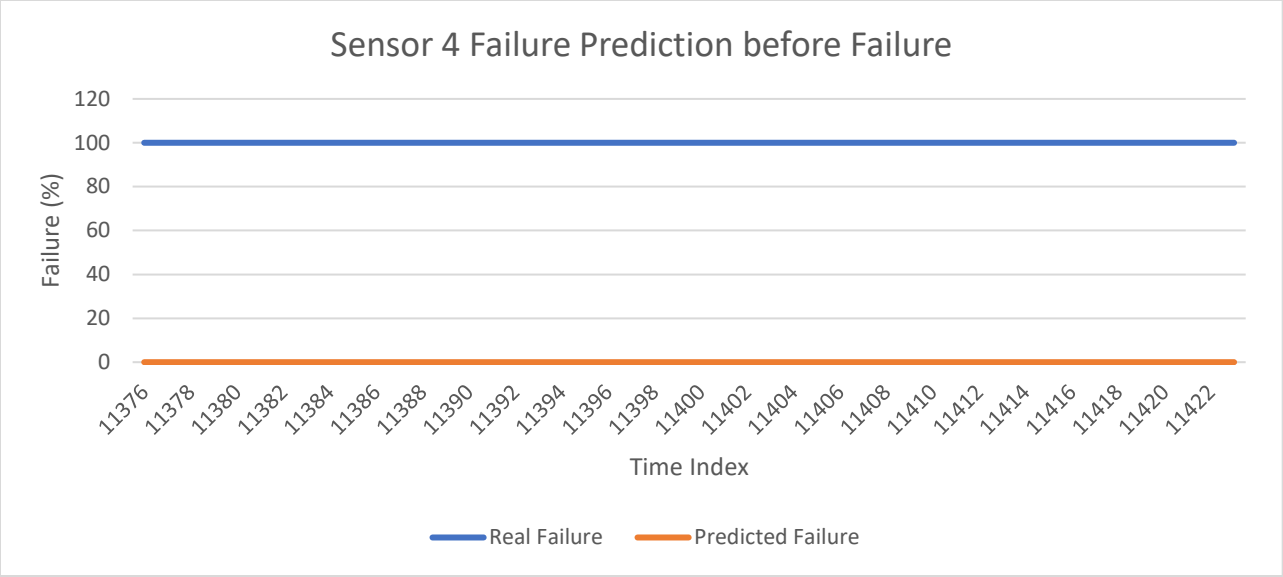


Figure 49 Sensor 4 Failure prediction before failure

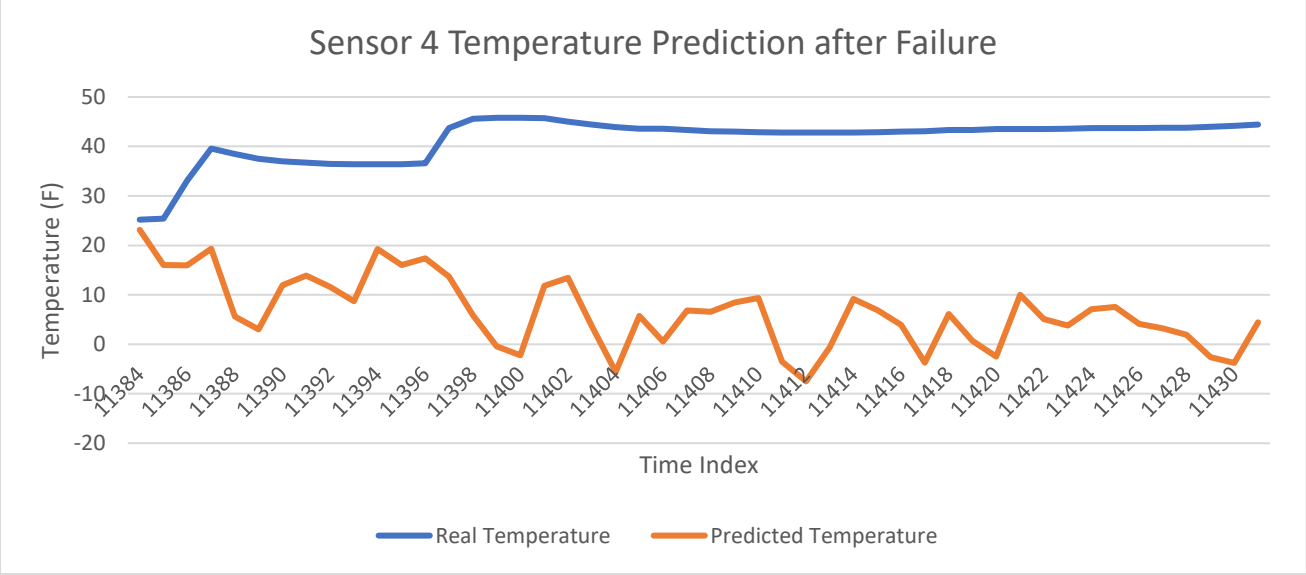


Figure 50 Sensor 4 Temperature prediction after failure

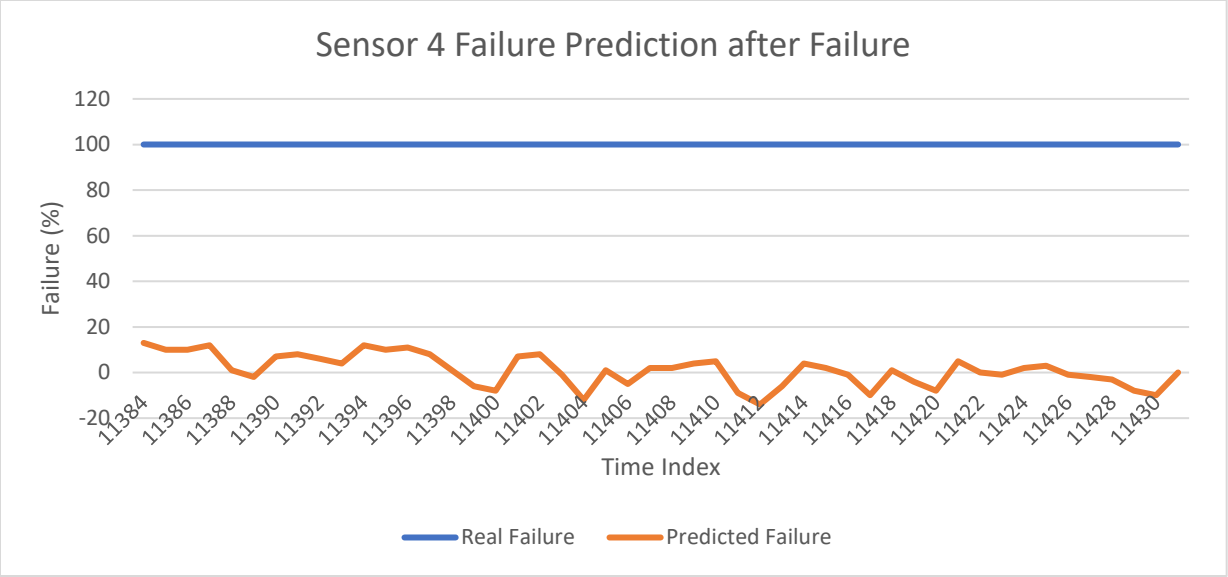


Figure 51 Sensor 4 Failure prediction after failure

Analysis

The temperature and failure percentage data are given in the Figure 47. The sensor fails at the time index of 11376. As shown in Figure 48 & 49, the model cannot predict the temperature of 1 day in future when the failure happened. As a result, it cannot predict the failure by using the data of past week. For this unit, failure happens suddenly without giving any information in the data of the previous week. As shown in the Figure 50 & 51, the input data was shifted by 4 hours (8 data points) to predict the temperature. The model given the *4 hour shifted* input data is able to predict failure with less than 20 confidence score.

8.6.5 Sensor 5 Results (Figure 52-56)

The raw temperature and failure data are given in the Figure 52. As shown in the figure, failure occurred at the data point 6503.

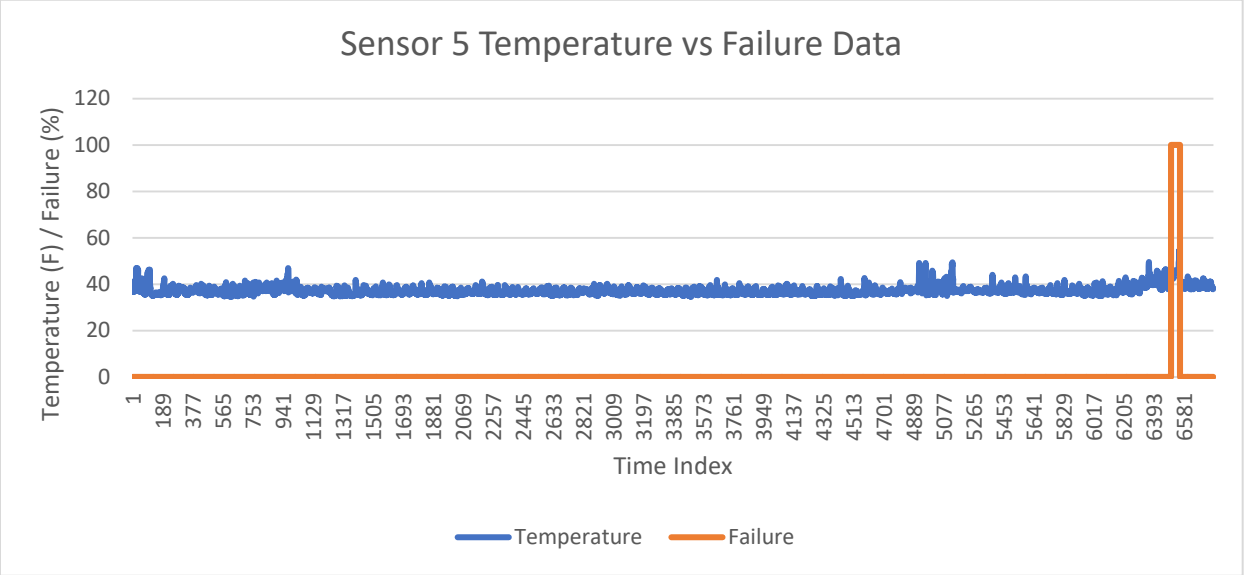


Figure 52 Sensor 5 Temperature vs Failure Data

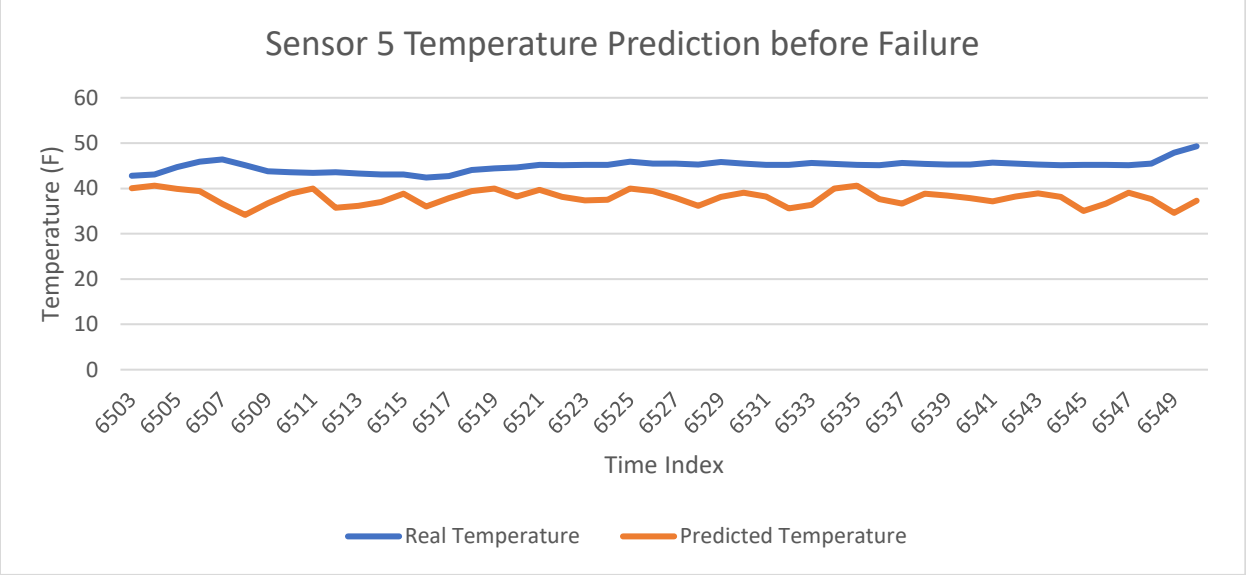


Figure 53 Sensor 5 Temperature prediction before failure

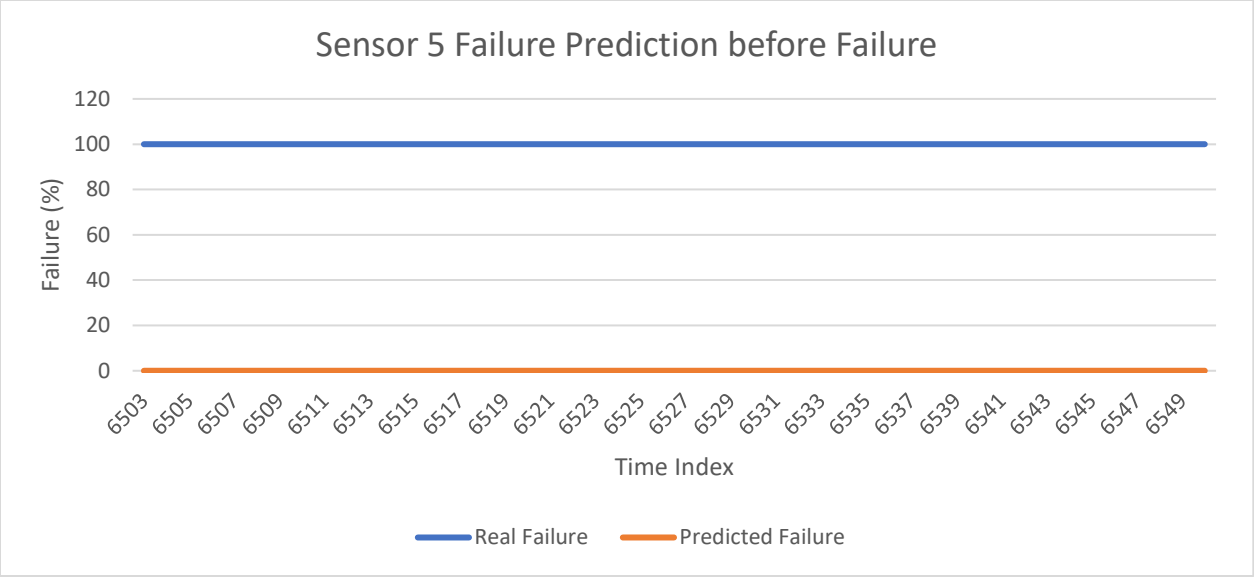


Figure 54 Sensor 5 Failure prediction before failure

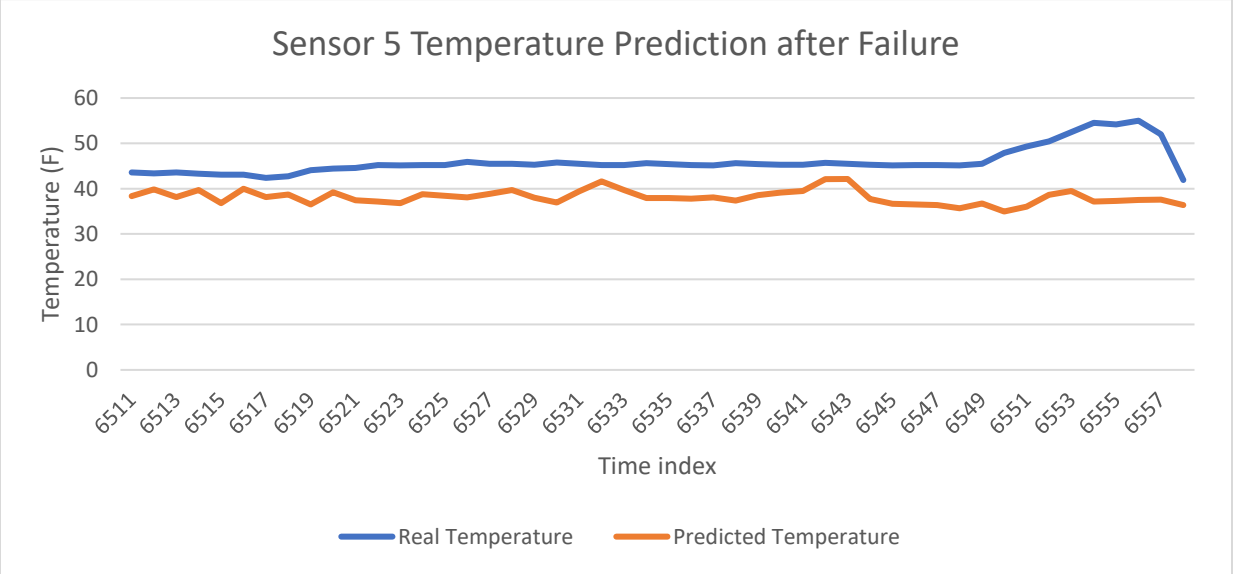


Figure 55 Sensor 5 Temperature prediction after failure

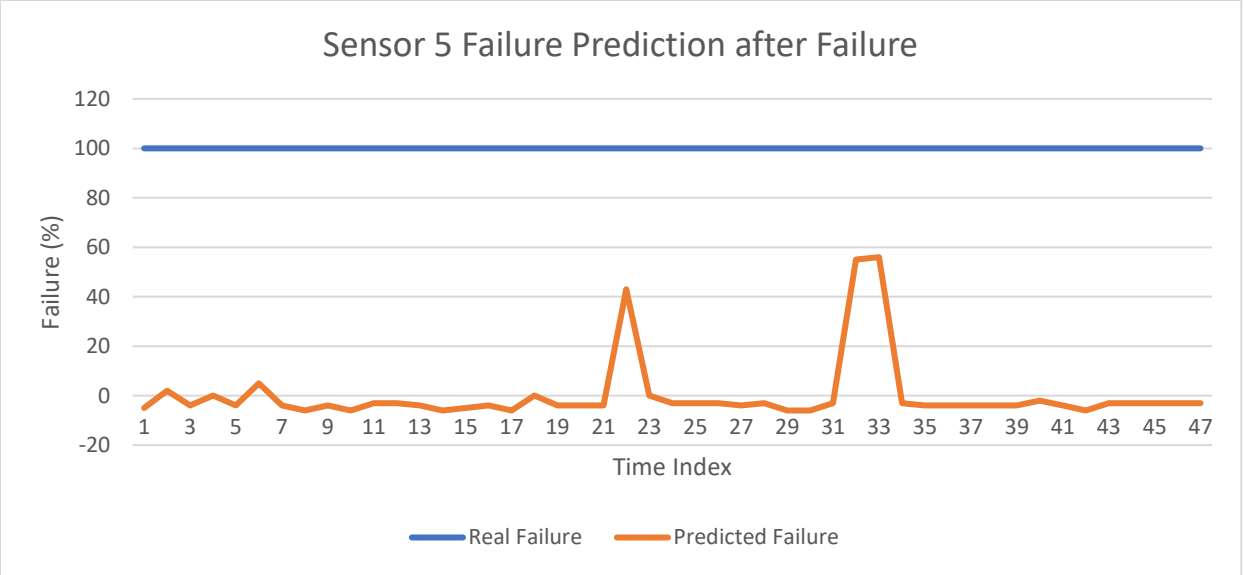


Figure 56 Sensor 5 Temperature prediction after failure

Analysis

The temperature and failure percentage data are given in the Figure 52. The sensor fails at the time index of 6503. As shown in Figure 53 & 54, the model cannot predict the temperature of 1 day in future when the failure happened. As a result, it cannot predict the failure by using the data of past week. For this unit, failure happens suddenly without giving any information in the data of the previous week. As shown in the Figure 55 & 56, the input data was shifted by 4 hours (8 data points) to predict the temperature. The model given the *4 hour shifted* input data is able to predict failure with the 50% confidence score at the time index of 22 and 32. The time index of 1 is the data point of 6511 as shown in the Figure 55.

8.6.6 Sensor 6 Results (Figure 57-61)

The raw temperature and failure data are given in the Figure 57. As shown in the figure, failure occurred at the data point 3051.

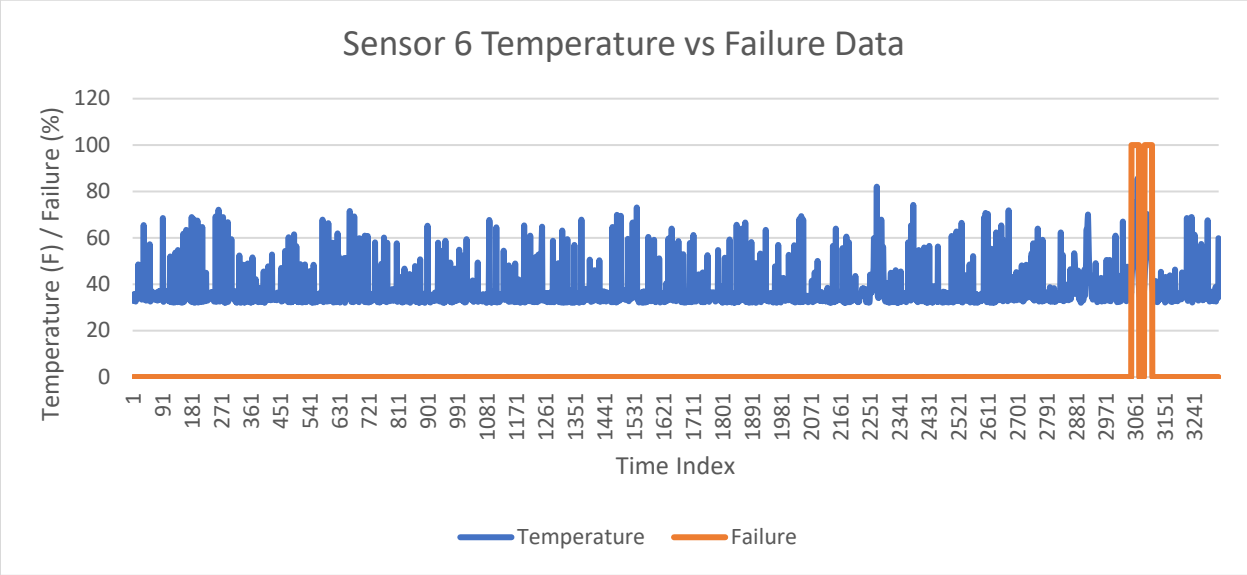


Figure 57 Sensor 6 Temperature vs Failure

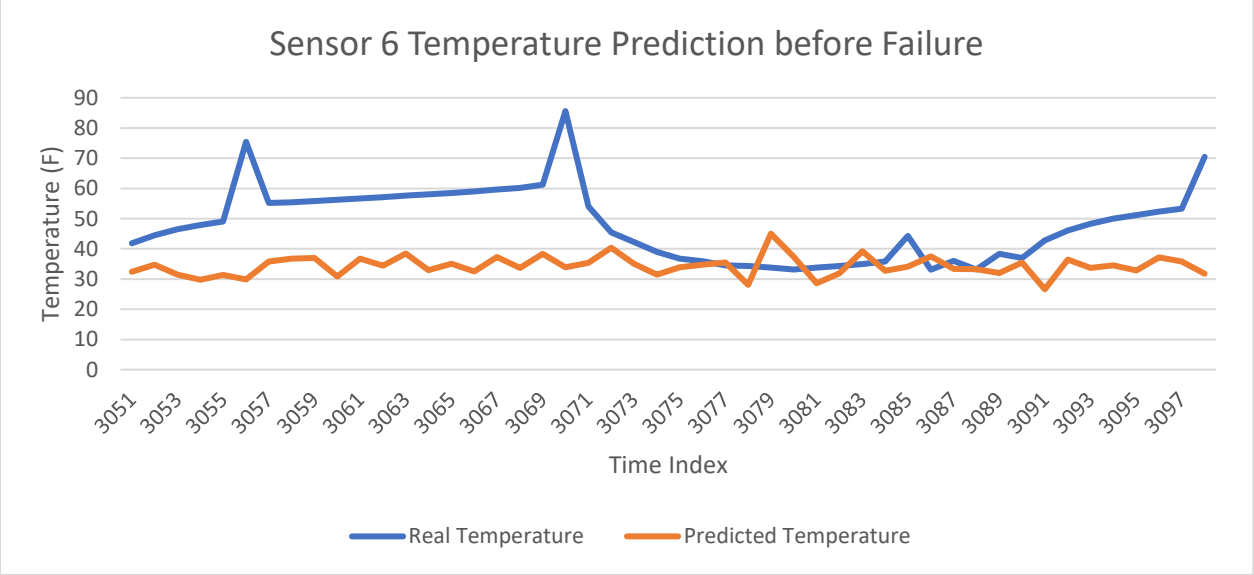


Figure 58 Sensor 6 Temperature prediction before failure

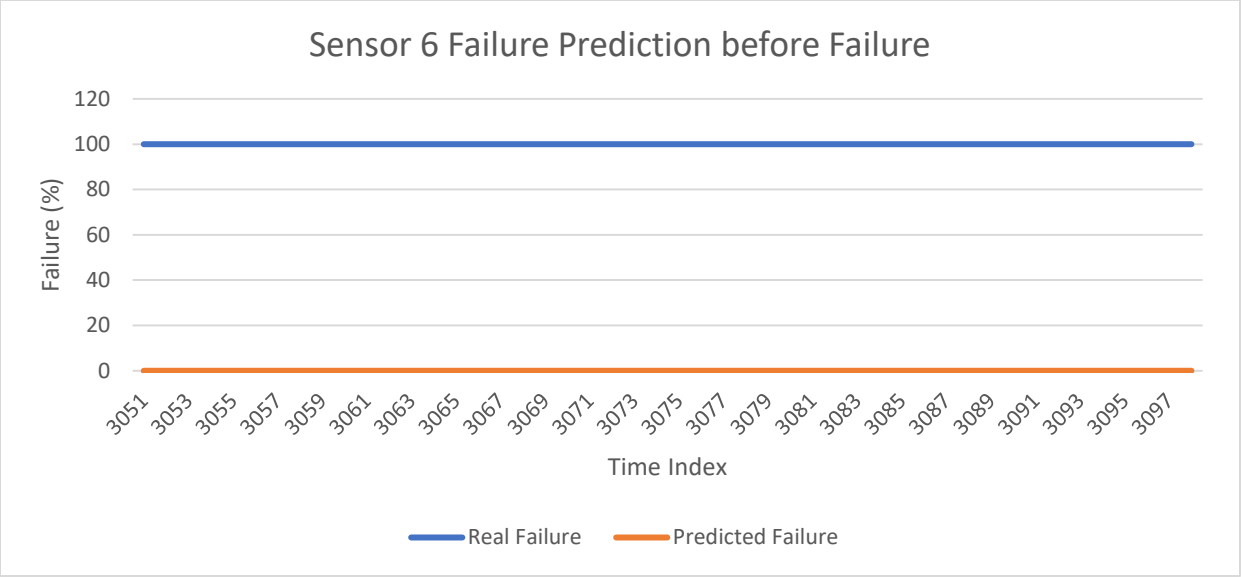


Figure 59 Sensor 6 Failure prediction before failure

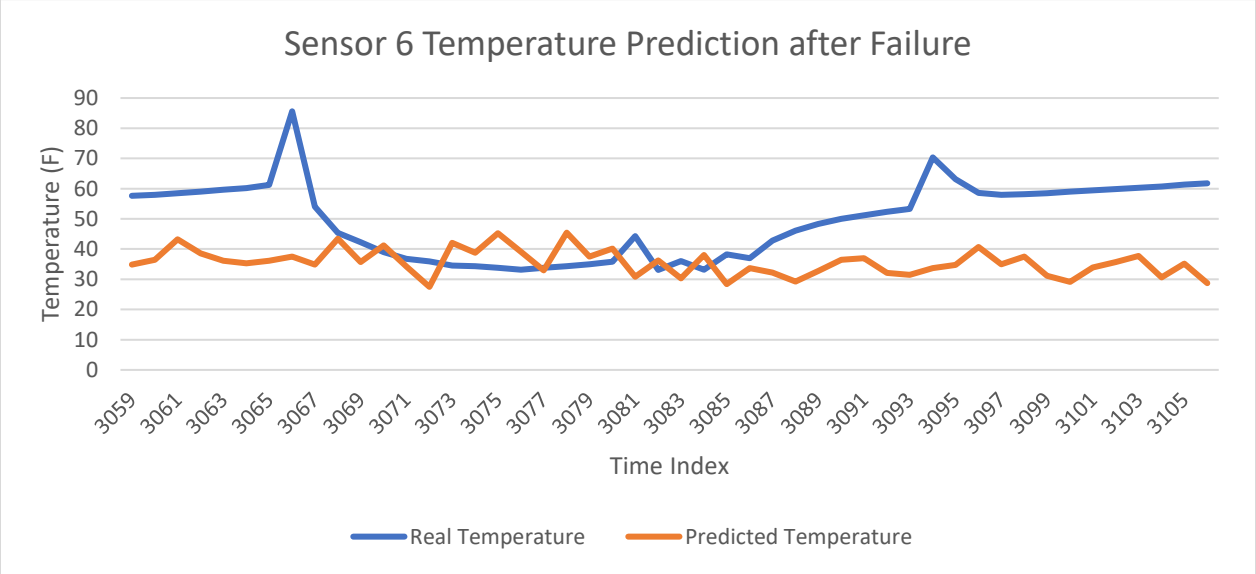


Figure 60 Sensor 6 Temperature prediction after failure

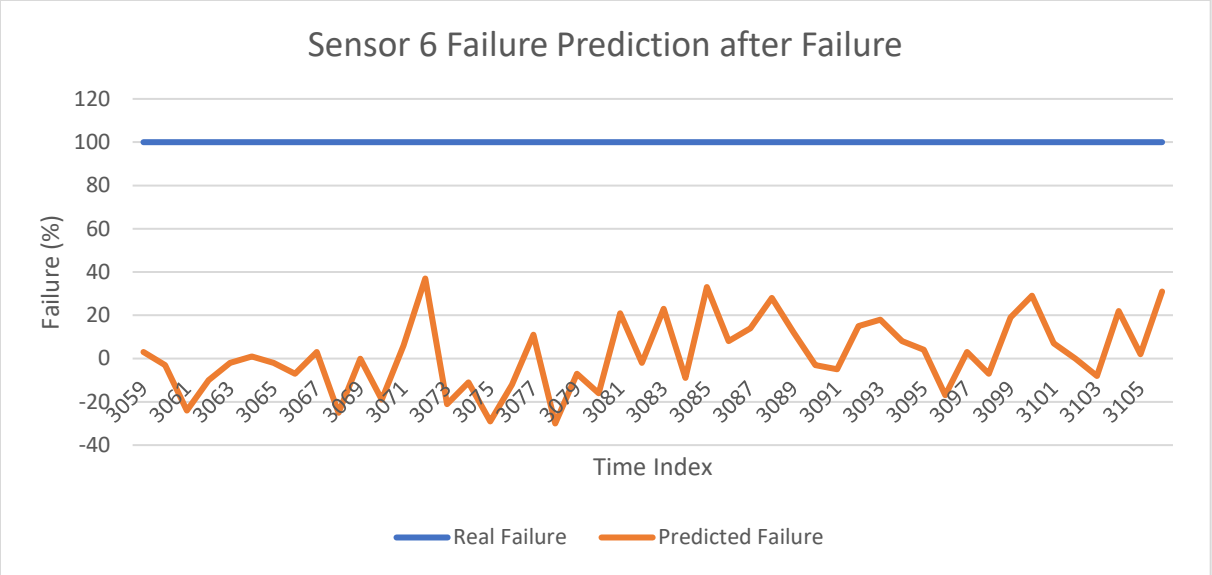


Figure 61 Sensor 6 Failure prediction after failure

Analysis

The temperature and failure percentage data are given in the Figure 57. The sensor fails at the time index of 3051. As shown in Figure 58 & 59, the model cannot predict the temperature of 1 day in future when the failure happened. As a result, it cannot predict the failure by using the data of past week. For this unit, failure happens suddenly without giving any information in the data of the previous week. As shown in the Figure 60 & 61, the input data was shifted by 4 hours (8 data points) to predict the temperature. The model given the *4 hour shifted* input data is not able to predict failure.

This test gives useful insight into the failure prediction using the past 1 week of data before the failure. Based on the results, I can say that there are two different types of failure. The first is a sudden failure, which happens without any changes in the temperature before the failure happens. In this situation, ML algorithm cannot predict the failure. The second type of failure is a gradual failure. As shown in the Sensor 3 data, the ML algorithm is able predict the failure. In the gradual failure, temperature rises over time before it reaches to the failure threshold. This behavior can be captured by the ML algorithm by training the model over large set of historical data. After testing the model with six different sensors, only sensor 3 showed gradual failure. All other sensors have sudden failure which is hard to detect using ML algorithm.

9 Conclusion

“Internet of Things” enabled systems can provide deep insights about system efficiency and performance. However, the time interval of data collection is important to analyze system behavior. Sensors can help to monitor the performance of electro-mechanical systems such as a refrigerator, thermostat, oven etc.

In this paper, I attempted to use machine learning to predict refrigerator temperature and failure based on historical temperature data. The temperature profile of the sensor depends on the type of system where it is installed. The results show that grouping sensors based on temperature profile can help to predict temperature and failure of one sensor from the trained model of another sensor from the same group. After careful observation of the results of many sensors, it has been observed that refrigerators have two types of failures. The first type of failure is sudden failure where temperature rises above average range suddenly without showing any characteristic of failure. In this scenario, the algorithm cannot predict failure because only temperature data is not sufficient. The second is gradual failure where temperature starts rising incrementally over time. In this scenario, the LSTM machine learning algorithm can find a signature of failure with the help of historical data.

10 Future Work

A linear regression is not able to find the temperature and failure prediction because the temperature inside the cooler follows different defrost cycles over the day. During this defrost cycles, the temperature rises for first few hours and then it drops back to lower temperature. A polynomial regression with higher degrees may provide temperature and performance prediction. A higher degree polynomial equation may find the periodical pattern of the temperature to make better performance prediction. A higher degree polynomial takes large amount of time to fit the data for prediction with the normal CPU. A machine with a high processing power can reduce the time to generate results. This approach was not used due to lack of high processing power machine and lack of time.

A Support Vector Regression is a regression algorithm which can be used for continues values like time series temperature data. In a simple regression, the algorithm attempts to reduce error. While in SVR, the algorithm tries to fit the error within a threshold. For this project, the error is not important, but the prediction of temperature peak value is very important. The peak values indicate the failure of a cooler. So, A Support Vector Regression may also provide a performance prediction. This approach was not followed due to lack of time and to explore the other methods to predict the performance of a refrigerator using ML algorithms.

In this project, input data of 1 week was used to find the temperature and failure probability for the next day. One day's worth of input data was not tested to predict the failure. It may or may not provide better insight into failure prediction with the data of 1 day. The places where the restaurant follows a daily schedule, training the model by 1 day of data may give better prediction. This approach was not followed because the available sensor data is from a restaurant which follows weekly schedules of operation.

In this project, the LSTM model is trained with neuron count of 256. The neuron count is directly related to the computation time of the model. To generate more results for different sensors, a higher number of neuron count was not used to train the model due to limited time. The LSTM model with neuron count 336 can be used for training the model. The RMSE value can be used to compare the results of higher neuron count with the lower neuron count values. This approach was not followed because of limited time available. The neuron counts 336 will take large amount of time to train the model with normal configuration CPU.

In this project, the sensor prediction was performed by training the historical data of that sensor. The Sensor Group approach can be used to make predictions with the data of another sensor. In this method, the sensors can be divided into groups based on the size of cooler, temperature and humidity range, equipment type etc. In this approach, the historical data of another sensor can be used to predict the behavior of a sensor which does not have any historical data. This approach was performed to show the potential of group-based prediction only.

To improve the performance of the algorithm, power consumption data of the refrigerator can provide more detail into electrical system behavior. The electrical characteristics of the refrigerator changes faster than the temperature inside the refrigerator. If the power is not consistent, the cooling system will fail. This cooling system failure event can provide better prediction in the case of sudden failure of the refrigerator. It can also use door Open/Closed events, food temperature and amount of empty space to make better predictions of the temperature and the failure. In this study, the door open/close event and other daily operations were not considered because I did not have this data. The quality of data could be improved by taking into consideration the activities performed by the people inside the kitchen.

11 References

Adafruit 1991. (n.d.). Retrieved from Mouser: https://www.mouser.com/ProductDetail/Adafruit/1991?qs=N%2F3wi2MvZWCAI8hF9jA4vQ%3D%3D&gclid=Cj0KCQiA597fBRCzARIsAHWby0GAkRgKxBhb9yCWkTJo2S6BAROrtCVGaPJwevicnqWgDGQrGqUd7HEaAlTHEALw_wcB

MikroElektronika MIKROE-1797. (n.d.). Retrieved from Mouser Electronics: https://www.mouser.com/ProductDetail/MikroElektronika/MIKROE-1797?qs=IV%252bl53bggh8OVl%2FvF4gESg%3D%3D&gclid=Cj0KCQiA597fBRCzARIsAHWby0GajDEA6t_VLpBM8xuFQHcf6QSGxDRgc3ZCNa_V4pm2LZwt46ImVtcaAnkUEALw_wcB

7 Types of Regression Techniques you should know! (n.d.). Retrieved from https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/

alliance, Z. (2012). *Zigbee Specification.*

Amazon SageMaker. (n.d.). Retrieved from https://aws.amazon.com/sagemaker/

Antonio Gulli, S. P. (2017). *Deep Learning with Keras.* Packt Publishing Ltd.

AWS IoT. (n.d.). Retrieved from The broadest and deepest portfolio of IoT services: https://aws.amazon.com/iot/

Barnes, J. (2015). *Azure Machine Learning.* Redmond, Washington: Microsoft Press.

Basics of PCB. (2018, 1 03). Retrieved from Circuit digest: https://circuitdigest.com/tutorial/basics-of-pcb

Eclipse MQTT Paho. (n.d.). Retrieved from https://www.eclipse.org/paho/

Get Started with TensorFlow. (n.d.). Retrieved from https://www.tensorflow.org/tutorials/

How DeepAR Works. (n.d.). Retrieved from https://docs.aws.amazon.com/sagemaker/latest/dg/deepar_how-it-works.html

Instrument, N. (2012, June 12). *Five Factors to Consider When Implementing a Wireless Sensor Network (WSN).* Retrieved from http://www.ni.com/white-paper/10789/en/

Keras. (n.d.). Retrieved from Keras: https://keras.io/

K-Nearest Neighbor. (n.d.). Retrieved from http://www.data-machine.com/nmtutorial/classificationproblemknearestneighboralgorithm.htm

Lithium Ion Battery - 1Ah. (2018). Retrieved from Sparkfun: https://www.sparkfun.com/products/13813

Maxim Integrated MAX31855 Cold-Junction Compensated Thermocouple-to-Digital Converters. (n.d.). Retrieved from https://www.mouser.com/new/maxim-integrated/maximmax31855/?gclid=CjwKCAjw_IPcBRAjEiwAl44QkcW77TyQUkAmu8HmoYfYk9YzJchydYpz04FchfhdcdndhwgfnGoqZBoCMW8QAvD_BwE

MQTT. (n.d.). Retrieved from http://mqtt.org/

MQTT 101 – How to Get Started with the lightweight IoT Protocol. (n.d.). Retrieved from http://www.eclipse.org/community/eclipse_newsletter/2014/october/article2.php

Next Big Things in IoT predictions for 2020. (n.d.). Retrieved from https://www.itproportal.com/features/next-big-things-in-iot-predictions-for-2020/

PICAXE 28X2 Microcontroller (28 pin). (2018). Retrieved from Sparkfun: https://www.sparkfun.com/products/9195

Polynomial Regression. (n.d.). Retrieved from https://towardsdatascience.com/polynomial-regression-bbe8b9d97491

Raspberry Pi Zero W. (n.d.). Retrieved from https://www.raspberrypi.org/products/raspberry-pi-zero-w/

Refrigeration and Food Safety. (n.d.). Retrieved from https://www.fsis.usda.gov/shared/PDF/Refrigeration_and_Food_Safety.pdf

Type T Thermocouple. (n.d.). Retrieved from https://www.thermocoupleinfo.com/type-t-thermocouple.htm

Understanding LSTM Networks. (2015, August 27). Retrieved from colah's blog: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Unsupervised Learning. (n.d.). Retrieved from https://www.mathworks.com/discovery/unsupervised-learning.html

What is LoRaWAN specification? (n.d.). Retrieved from LoRa Alliance: https://lora-alliance.org/about-lorawan

What is TensorFlow? (n.d.). Retrieved from https://www.computerworlduk.com/open-source/what-is-tensorflow-how-are-businesses-using-it-3658374/