University of Nottingham
UK | CHINA | MALAYSIA

Faculty of Science and Engineering

Department of Electrical and Electronic Engineering

Report

Undergraduate Summer Research Internship

ACADEMIC YEAR 2021/2022

Student's Name	: Arfan Danial Bin Zainal Abidin
Student's ID	: 20284997
Supervisor's Name	: Marwan Nafea
Project Title	: Fruit Ripeness Identification using Electronic Nose
Contact Number	: 0192150612
Year of Study (2021/2022)	: Please tick (√)

Year 1	(✓)
Year 2	( )
Year 3	( )

Programme : Please tick (√)

Electrical and Electronic Engineering	(✓)
Mechatronic Engineering	( )

Student's Signature : Arfan.

1.0 Introduction

Electronic Nose can be described as a device that detects and recognizes odours or aromas. This is made possible by utilizing gas sensors and classification models. The gas sensors output combined can create a distinct digital pattern. The classification model will then learn the patterns and classify the aroma. The human sense of smell is far less sensitive when compared to other animals. The electronic nose is made to mimic the human nose but with far better sensitivity and without emotional biases. The sensitivity of the nose varies from person to person [1]. The human's ability in recognizing odour is said to be heavily link with emotions [1]. This could lead to inconsistent identification of the odour. In addition, sicknesses can also have a huge impact on a person's sense of smell. The uses of e-noses have risen a lot since more new application are discovered. These include monitoring quality and freshness of food product.

The purpose of this research is to create an e-nose system capable of determining the ripeness of a fruit. The ripeness of a fruit is one of the most important factors to know when determining the fruit's shelf life. Many researchers denote that the aroma released by the fruit vary at different ripeness level [1]. The e-nose created utilizes MATLAB for data gathering of the fruits. Due to availability, the chosen fruits are bananas, mangoes, and tomatoes. The ripeness levels in consideration are unripe, ripe, and overripe. With the help of MATLAB's add-on application, the building of classification models for identifying the fruits were made easily as it is beginner friendly. A classification model utilizes one of the possible machine learning algorithms such as decision tree, support vector machine, and many more.

2.0 Objectives

The objectives of the research are as follows,

1. Assemble a gas chamber of the e-nose for data collection of the fruits
2. Collect data on three different fruits at varying ripeness level
3. Create the best possible classification model for the e-nose system by using the data collected

3.0 Methodology

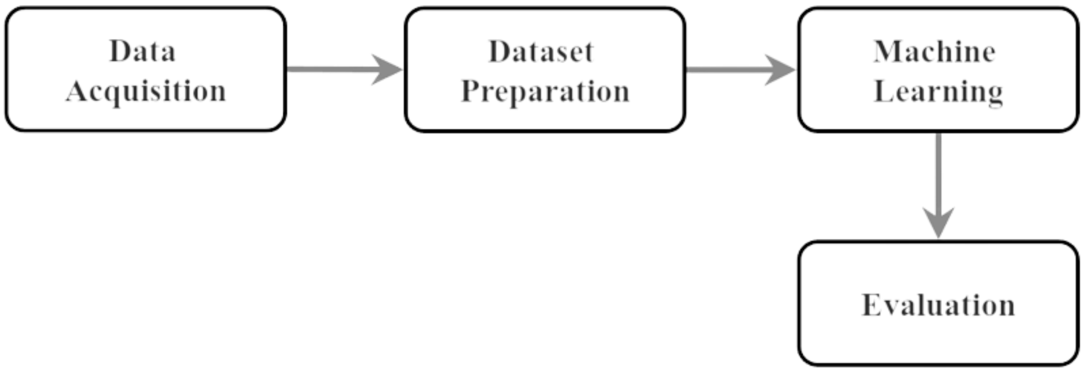


Figure 1: System Workflow

Figure 1 demonstrates that there are four stages for this e-nose system. The first stage focuses on acquiring data from selected fruits. Second stage will then prepare datasets according to the data collected from the gas sensors. The third stage will be machine learning, where training and testing of classification models will happen using the prepared datasets. Final stage will be the evaluation of the classification model's performance for each training and testing phase.

4.0 Data acquisition

4.1 Working principle of MQ Gas sensors

All the sensors used are metal oxide semiconductor gas sensors. More specifically, each MQ-sensor uses a thin film of tin oxide. This metal oxide is a n-type semiconductor. This means in the increase concentration of oxidizing gases like oxygen, the number of electrons will decrease on the surface which leads to a higher resistance in the semiconductor. This will then reduce current flowing through the sensor. Hence, a lowering the analogue voltage value as depicted in Figure 2.

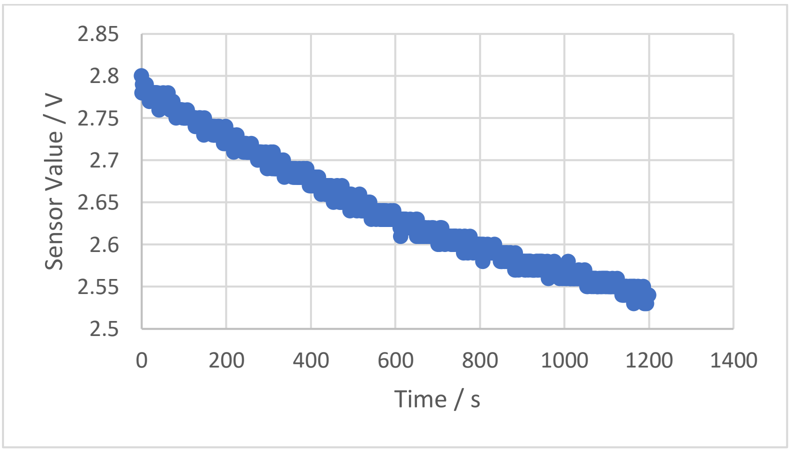


Figure 2: Increase in oxidizing gas

Otherwise, in the increases presence of reducing gases, the number of electrons on the surface of the oxide layer will increase which lowers the resistance in the semiconductor. Thus, increasing the current flowing through the sensor. This will then generate an analogue voltage value from the sensor to the microcontroller. The increase in the concentration of the reducing gas, the results in a larger analogue value until a steady-state value as illustrated in Figure 3.

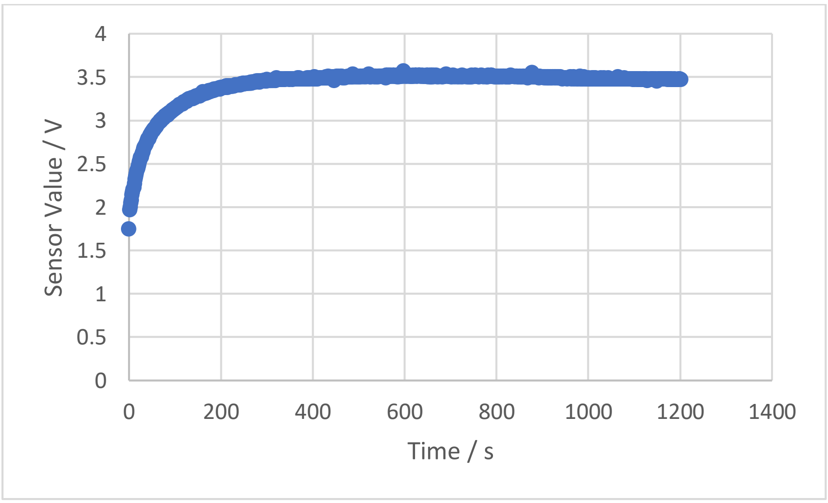


Figure 3: Increase in reducing gas

4.2 Circuit schematic

An Arduino MEGA microcontroller to cater for the use of many analogue pins by the gas sensors. The system hardware utilizes eleven MQ-series gas sensors with each having different range of target gases. Information on the gas sensors connections and target gases are shown in Table 1. The gas sensors are placed on the same wall of the container as illustrated in Figure 5 (b). There were two power supplies, the Arduino MEGA and the 5V step down power supply, powering the components. This gives the gas sensors sufficient voltage of 3.2V and 75 mA current. For each sample data, the DHT22 sensor will be used to determine the average humidity and temperature with an accuracy of ± 5 % and ± 0.5 °C, respectively. This will only be used as reference and not as input data for the classification models. A 12 cm by 12 cm DC fan is used to regulate in the air in the gas chamber. This is very important for distributing the gases from the fruits evenly to all eleven sensors.

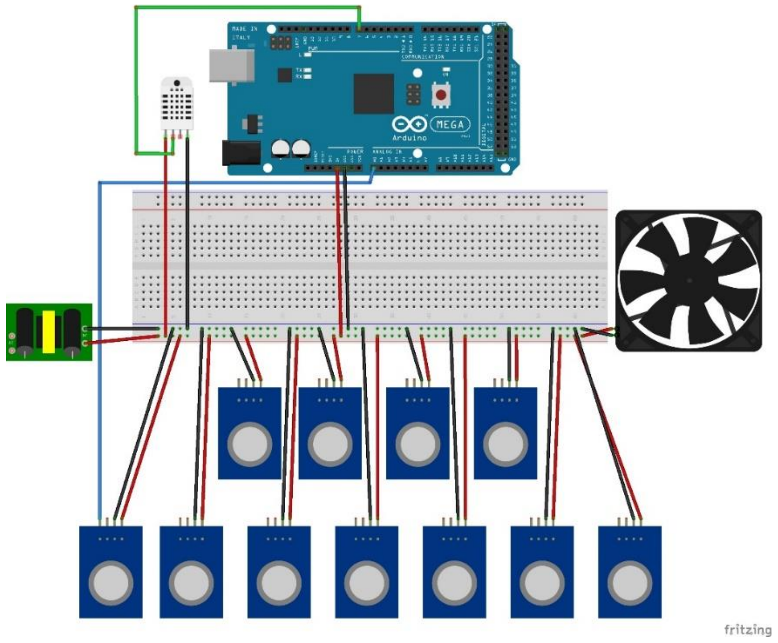
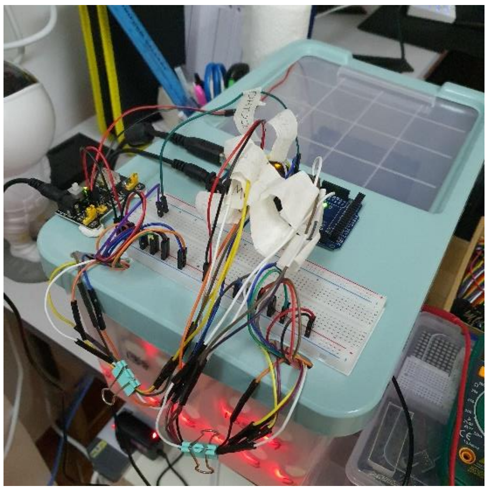
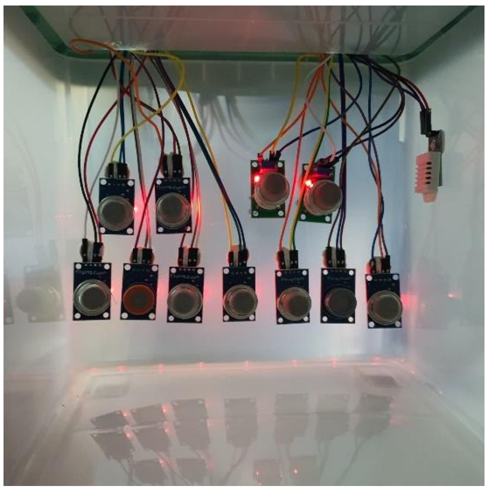


Figure 4: Circuit Schematic excluding Analog pins



(a) Position of Arduino and step down



(b) Position of sensors in chamber

Figure 5: Setup of the gas chamber

Table 1: Gas sensor Information

Gas Sensor Model	Target Gas	Analog Pin
MQ-2	Methane, Butane, Liquified Petroleum Gas, Smoke	A10
MQ-3	Alcohol, Ethanol, Smoke	A8
MQ-4	Methane, Compressed Natural Gas	A6
MQ-5	Natural gas, Liquified Petroleum Gas	A5
MQ-6	Liquified Petroleum Gas, Butane Gas	A2
MQ-7	Carbon Monoxide	A1
MQ-8	Hydrogen Gas	A0
MQ-9	Carbon Monoxide, Flammable Gasses	A9
MQ-135	Air Quality (CO, Ammonia, Benzene, Alcohol, smoke)	A7
MQ-136	Hydrogen Sulphide gas	A4
MQ-138	Benzene, Toluene, Alcohol, Acetone, Propane, Formaldehyde gas, Hydrogen	A3

4.3 Workflow

One of the disadvantages using the MQ-series gas sensors is that it requires to be operated at high temperature to avoid production of very noisy analogue signals. To operate at high temperatures the sensor must be warmed up just by supplying power to the gas sensors. Before every data gathering session, the gas sensors are given 15 minutes warm-up time. During warm-up, the gas chamber is left opened as depicted in Figure 7 (a). Do keep in mind that it is not recommended to warm up the sensors before every data collection of a fruit to avoid wasting time. Warming up the sensor is only necessary once in the beginning of many continuous data collection of fruits.

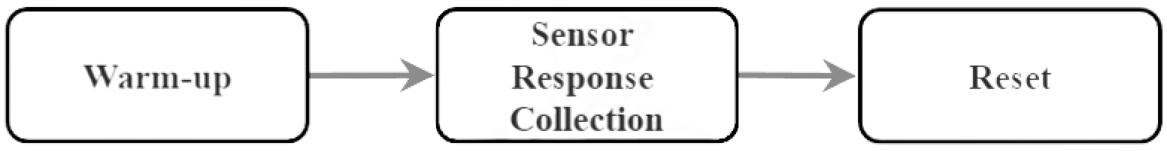
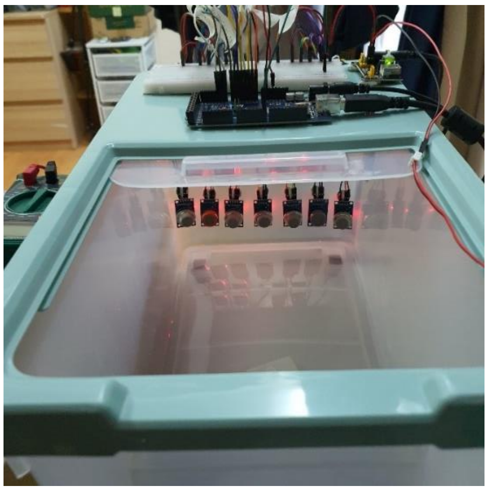


Figure 6: Data Acquisition Workflow

After warming up the sensors, a script in MATLAB is executed containing instruction for the microcontroller. A delay of 15 seconds is given before actual collection of sensor data to give time for placement of a fruit on a Teflon sheet that around 4 cm from the wall of gas sensors. The purpose of the Teflon sheet is to not allow odour of fruits to stick to the floors of the gas chamber for long periods of time [2]. Then, the slidable lid of the chamber is closed as shown in Figure 7 (b).


(a) Opened gas chamber

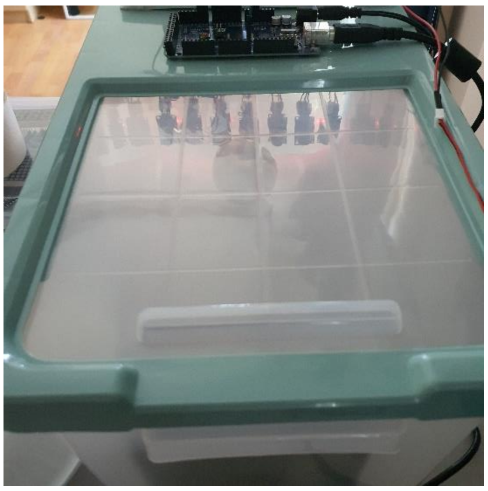
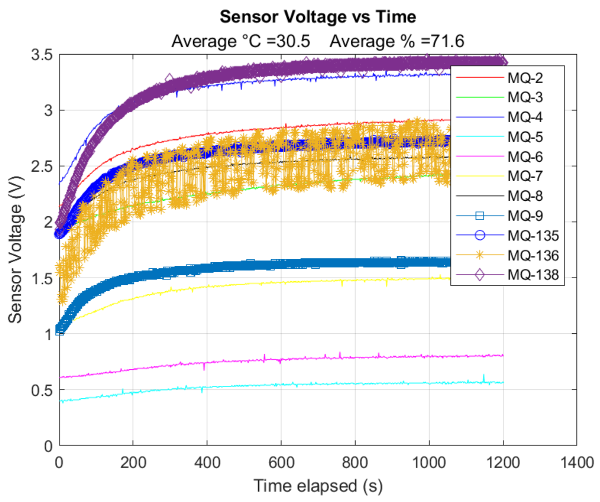
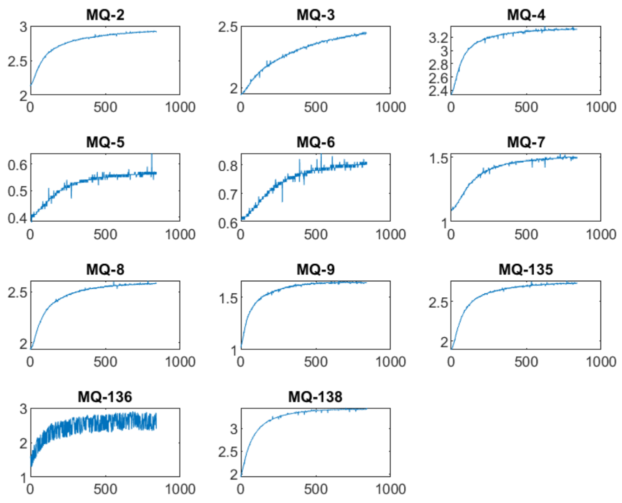

(b) Closed gas chamber

Figure 7: State of gas chamber

During the data collection of a fruit, the command window in MATLAB will print the time, number of data collected by each sensor, temperature, humidity, and each gas sensor’s current analogue value. Value from each sensor will be collected every 1.4 s for 20 minutes. After that, a summary will be shown containing the number of data collected by each sensor, average humidity, and temperature of the when collecting data of a fruit. The collected data will be export to an Excel worksheet for processing later. In addition, a combined and individual graph of the sensor response over time will be plotted automatically. An example of the graph is shown in Figure 8.



(a) Combined



(b) Individual

Figure 8: Graph of sensor response example

After every data collection of a fruit, resetting of the gas chamber is necessary before starting data collection on another fruit. The purpose of resetting is to remove the gas release from previous fruit as much as possible. This is to assure that the gas sensors are back at their baseline level. This is important to get an accurate reading from the gas sensors of a fruit. To initiate reset process, the fruit must be removed from the gas chamber and the slidable lid must be open. The fan is then placed on lid of the gas chamber while the chamber is open as illustrated in Figure 9. It is important to take note of the direction of the fan to ensure the fan to blowing air out of the chamber instead of into the chamber. The resetting process lasted for 15 minutes.

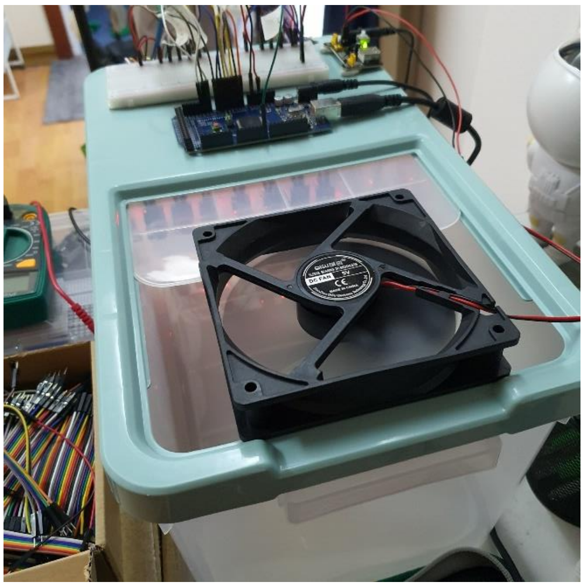


Figure 9: Reset process setup

4.4 Data Preparation

A different MATLAB script will be executed for this stage. Data preparation is necessary for determining what to include in a dataset that will be used to train and test a classification model. From Figure 10, there are four stages to this process. Raw data will be imported from an Excel worksheet to be process. In the processing phase, the raw data can either be passed on directly for feature extraction or undergo some signal pre-processing.

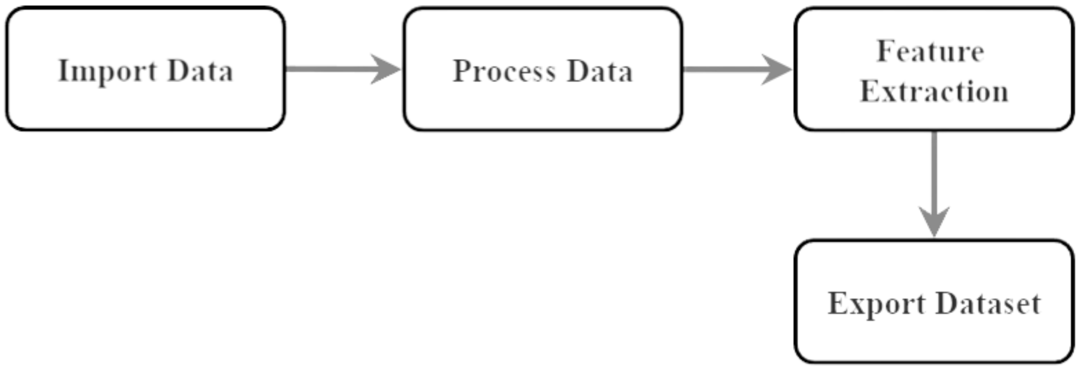


Figure 10: Flow of data preparation

4.4.1 Signal pre-processing

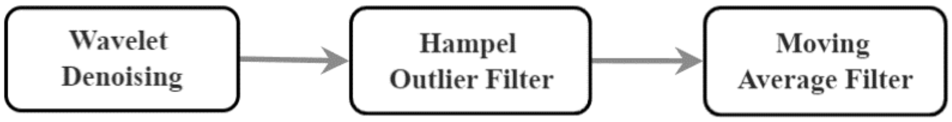
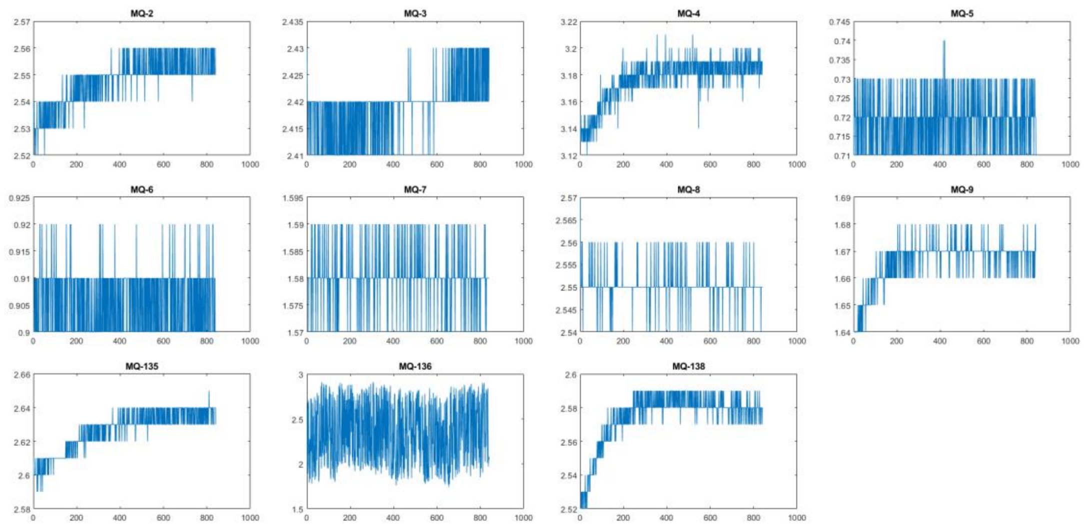
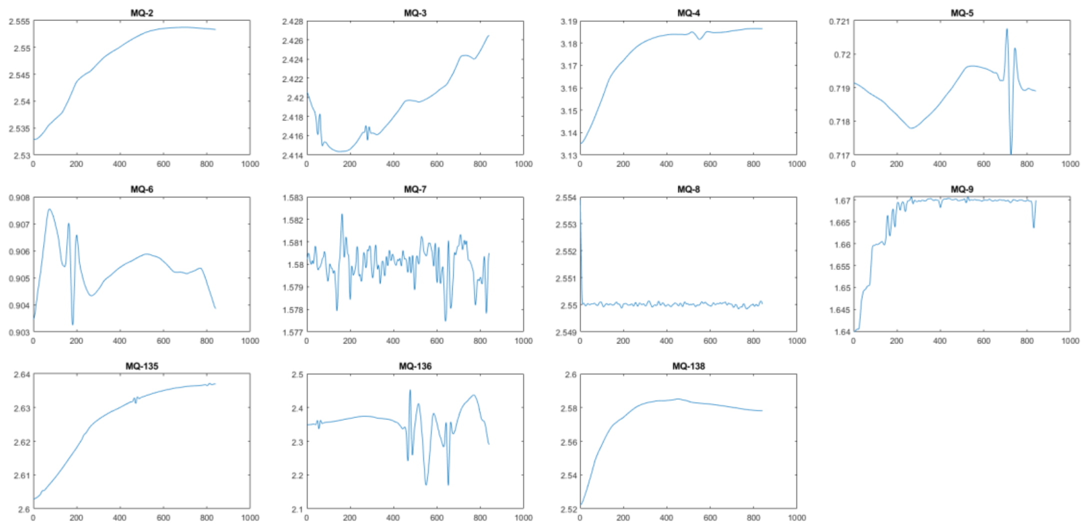


Figure 11: Steps of pre-processing

There are many signal pre-processing methods available in MATLAB. The purpose of signal pre-processing to refine the analogue signals generated by the gas sensors. This can be done by minimizing the noises in a signal and smoothening it. An example of signal pre-processing is depicted in Figure 12.



(a) Before



(b) After
Figure 12: Before and After signal pre-processing

Step 1: Wavelet denoising

This method is the 1st step out of 3 pre-processing methods used to refine the signals. This procedure consists of three stages as depicted in Figure 13. During decomposition stage, the signal undergoes wavelet transformation. This lets the signal to be divided into groups of coefficients at different frequency [3]. This is necessary to obtained important information relating to the characteristics of the signal.

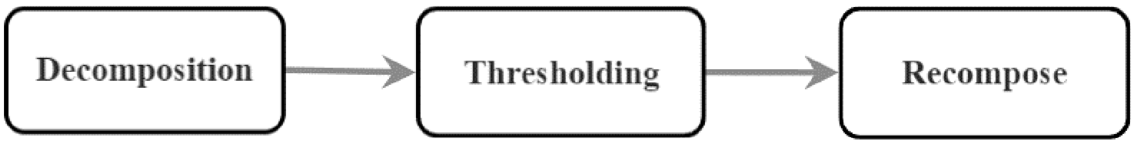


Figure 13: Workflow of wavelet denoising

The information obtained in the decomposition stage, demonstrate the behaviour of the signal at different frequency. Based on this, a suitable thresholding value can be decided which is then applied to the set of coefficients to remove the noise. The rule of thresholding used is named Empirical Bayes which assumes in a mixture model, the measurements have an independent prior distribution [4]. After thresholding the signal, the group of coefficients that was filtered is used to construct the final signal which is the product of wavelet denoising. An example of wavelet denoising is shown in Figure 14. When compared to the original Figure 12 (a), significant amount of noise has been filtered from the signals.

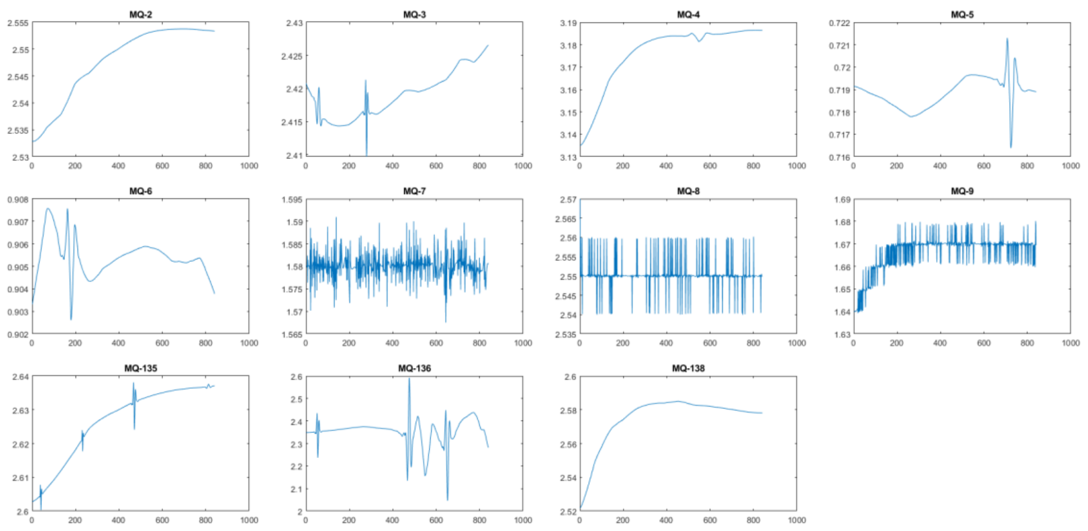


Figure 14: Example of wavelet denoising

Step 2: Hampel Outlier Filter

The denoising procedure alone is not enough to refine the signal. Another method used is the Hampel Outlier Filter (HOF), which based on the name itself, it is to remove most outliers that are present in a signal. The HOF decision making for filtering the outlier is based on the median and the median absolute deviation (MAD) estimator. The HOF version used was a moving window implementation [5].

Given a sequence x_1, x_2, x_n, x_4, x_5 a window of data is extracted with sample data x_n. Without specifying, MATLAB results to window data of length 3 which an example of it is x_1, x_2, x_n. From this window of data, the local median, m_i, and its MAD, S_k, are determined.
The formula to calculated S_k is shown below:

$$S_k = 1.4826 \; \times Median(\sum |X_i - m_i|)$$

The factor 1.4826 allows the MAD scale to make an unbiased estimation of the standard deviation of data samples [5]. $|X_i - m_i|$ is denoted as the absolute differences between a sample data, X_i, and m_i. Then compare each sample data in the window with $t \; \times S_k$, where t is the threshold value. The filter response works by if $|x_n - \; m_i,| > \; t \; \times S_k$, the data x_n is identified as an outlier and replaced with the local median value, m_i. Otherwise, the sample data remains the unchanged. The results of denoising with outlier removal is shown in Figure 15.

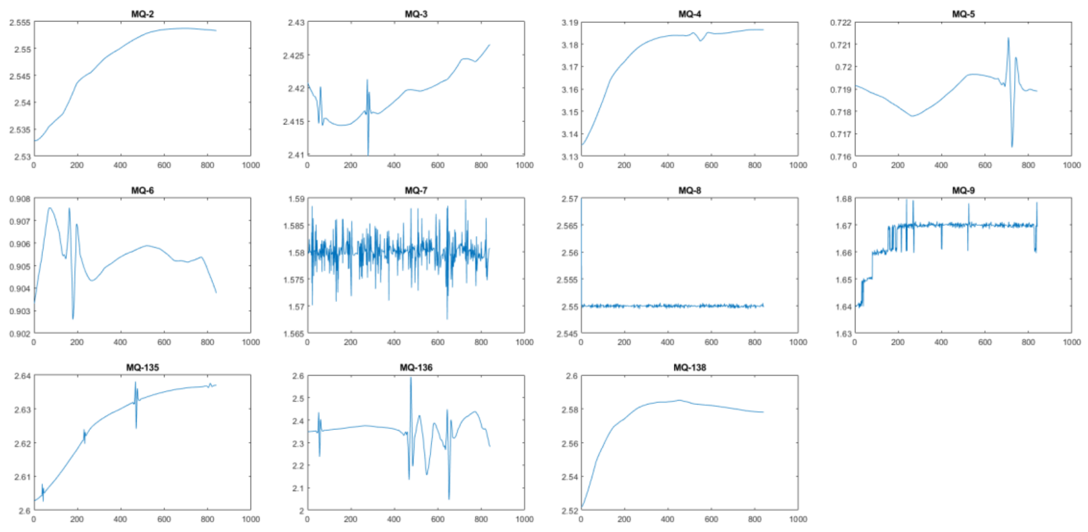


Figure 15: Denoising and Hampel Outlier Filter

<u>Step 3: Moving Average Filter</u>

The final method used in the signal pre-processing is the moving average filter. This method is used on the signal to smoothen it by further reducing the noise. Each point in the output signal of this filter is determined by averaging a number of points from the input signal. An equation can be used to represent this operation:

$$y[i] = \frac{1}{M} \sum_{p=0}^{M-1} x[i+p]$$

Where $y[\,]$ represents the output signal, $x[\,]$ is the input signal, and M is the number of points applied in the moving average filter. An example of this operation can be demonstrated below where a 5-point moving average filter is used, resulting in a 70 in the output signal:

$$y[70] = \frac{x[70] + x[71] + x[72] + x[73] + x[74]}{5}$$

Although in MATLAB, the number of points used in the filter is determined automatically based on the input signal to give the best possible results. There are a few variations of this filter. The one used in this research is a Gaussian-weighted moving average filter. This specific variation gives a smoother curve in the step responses compared to the default abrupt straight line of the moving average filter [6]. The result of this final step in the pre-processing was shown in Figure 12 (b).

4.4.2 Feature Extraction

The array of data which represents a signal cannot be fed directly to classification model. Instead, the input data to be fed to a classification model are features extracted from that signal. These features represent the characteristics of a signal. Below is the list of features that was extracted from a signal.

Mean

$$\bar{X} = \frac{\sum X}{N}$$

N = number of data

Root-mean square (RMS)

$$RMS = \sqrt{\frac{1}{N} \sum_i^N {x_i}^2}$$

N = number of data
x_i = each data

Standard deviation (Std)

$$Std = \sqrt{\frac{\sum_i^N (x_i - \bar{X})^2}{N}}$$

N = number of data
x_i = each data
$\bar{X}$ = mean

Peak to RMS ratio

$$Peak\ to\ RMS\ Ratio = \frac{||X||_\infty}{\sqrt{\frac{1}{N} \sum_i^N {x_i}^2}}$$

N = number of data
x_i = each data
$||X||_\infty$ = infinity-norm

Skewness

$$Skewness = \frac{\sum_i^N (x_i - \bar{X})^3}{\sigma^3 (N-1)}$$

N = number of data
x_i = each data
$\bar{X}$ = mean
σ = standard deviation

After the features have been extracted from a signal, it is then exported to be stored in an Excel worksheet. Features of signals from many different fruits are stored into one worksheet as datasets for machine learning later. One feature contains 11 sub-features due to signals from 11 gas sensors. For example, the mean of the signals collected from a fruit is 11 features.

4.5 Machine Learning

Import Datasets
Training
Testing

Figure 16: flow of machine learning

The datasets created from the feature extraction of signals are imported as predictor variables for the classification models. The datasets are split into 80% for training and 20% testing. A classification models utilizes a machine learning algorithm so that AI can be used to learn from the given dataset in training and attempt at correctly classify the fruit when given new data during testing. Furthermore, cross-fold validation of iteration 5 is used during training for each model to mitigate overfitting. Many machine learning models are tested in MATLAB’s classification learner application for the purpose of finding the best suited algorithm for the given datasets.

4.5.1 Decision Tree (DT)

Decision tree (DT) is a machine learning technique that follow a tree like structure as depicted in Figure 17. There are two types of nodes in a decision tree, which are the leaves and the decision nodes. The leaves are the final decision of the classification model which can be seen as the two bottom nodes in Figure 17. While the decision nodes are the intermediate nodes in the decision making leading to the leaves. Commonly at the decision node, a test occurs to determine the outcome associated with any of the subtrees given the attribute of an instance [7]. This instance is described as the beginning or root node of the tree which can be seen at the very top of the mapping in Figure 17.

When learning new dataset, the decision tree follows the concept of ‘divided and conquer’. This principle states that when all the output attributes values from several datasets are the same, a tree will form a left that represent the chosen output. Otherwise, an attribute value is chosen, which results in a branch from that node that leads to at least two distinct nodes, either of which can be a decision node or a leaf. Input datasets are sorted into groups to form a node based on the attribute category. This process will be repeated using the given dataset until a leaf is found.

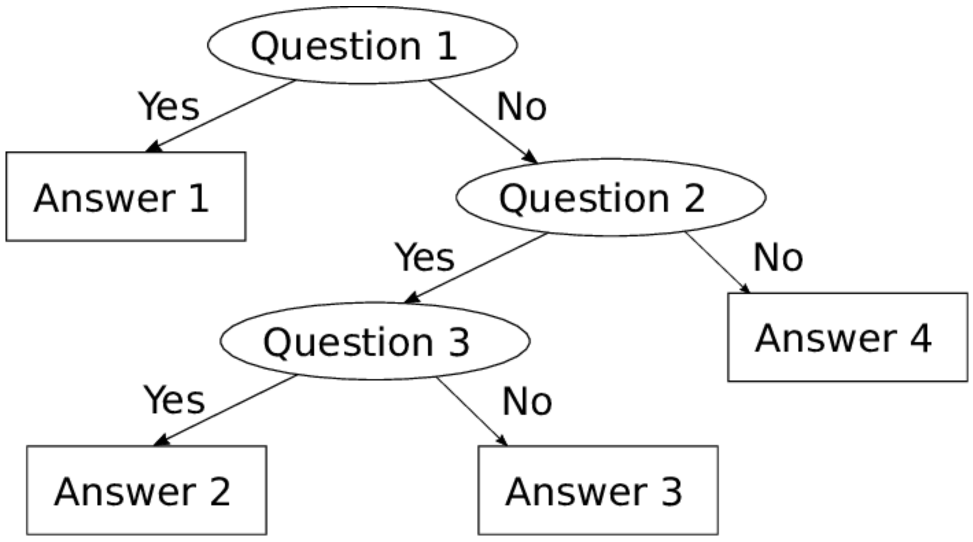


Figure 17: Decision Tree Algorithm [18]

4.5.2 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) working principle is based on creating a new variable from the given input dataset by combining them. The purpose of this is to fuse the input variables scores to create a new distinct composite variable which is the discriminant score. This analysis technique can be seen used for reducing the data dimension of the input as it reduces the input variable's p-dimension into a single-dimension line as illustrated in the right graph of Figure 18.

This procedure should result in each data group having a normal distribution of discriminant scores, but with the mean scores of each group being significantly different [9]. The discriminant score can be represented in the equation below:

$$D = w_1x_1 + w_2x_2 + w_3x_3 +, \dots\dots\dots\dots.. + w_nx_n$$

Where w is the suitable weight for data x to give the largest difference between the class mean discriminant score. In general, those data with the largest class mean difference will be assigned the largest weight while the smaller differences will result in assignment of the smallest weight [10].

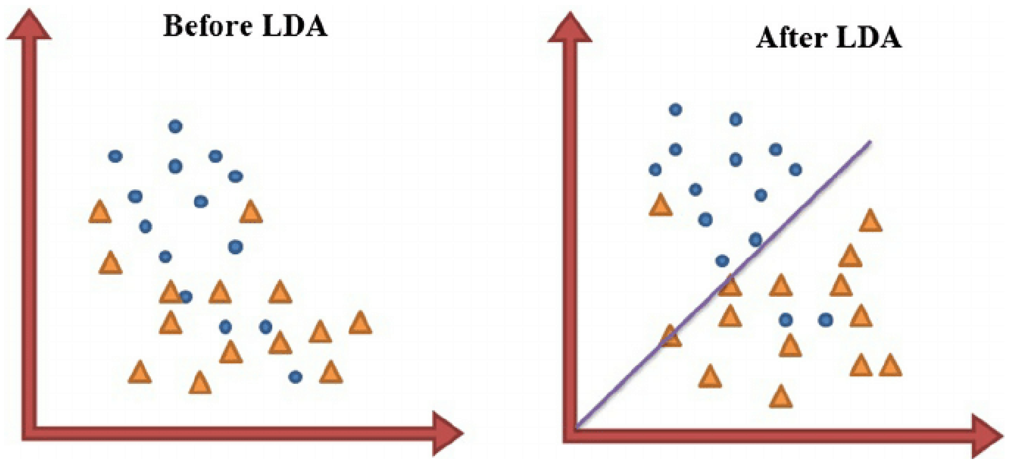


Figure 18: Principle of LDA [8]

4.5.3 Gaussian Naïve Bayes (GNB)

Gaussian Naïve Bayes (GNB) is a machine learning method that uses the Bayes' theorem as the foundation of classifying the input dataset into a pre-defined set of output groups given the information presented by the dataset's variables. GNB classifies by determining the conditional probability that, given the values of the dataset's variables, an input dataset results in a particular output under the presumption that each variable is class-conditionally independent [12]. Finding the conditional probability can be express as an equation as shown below:

$$P(x|y) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)}$$

Where x is the input data, y is the class output, μ is population mean, and σ is standard deviation. A simple model can be created under the assumption that the data are characterized by a Gaussian distribution that does not consider the covariance among dimensions as illustrated in Figure 19 [11]. The input data, x, is assigned to a class that results in the highest possible probability.

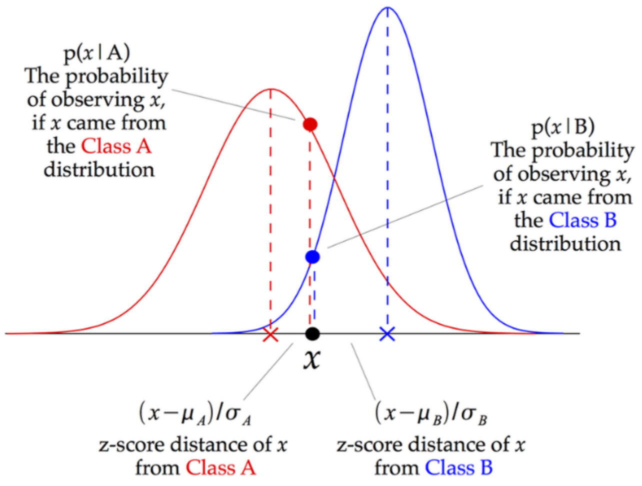

p(x | A)
The probability of observing x, if x came from the Class A distribution
p(x | B)
The probability of observing x, if x came from the Class B distribution
x
(x−μ_A)/σ_A
z-score distance of x from Class A
(x−μ_B)/σ_B
z-score distance of x from Class B

Figure 19: Working principles of GNB [11]

4.5.4 Linear Support Vector Machine (LSVM)

Support Vector Machine (SVM) classify data by constructing hyperplanes that best divides the input datasets. There are two variants of SVM which are linear and non-linear. LSVM only uses hyperplane while non-linear SVM uses planes in higher dimensions. Hyperplanes can be described as the straight line that divides and predict the data's class. This line is as $wx + b = 0$ as depicted in Figure 20. The aim should be to classify the data further away from the hyper-plane while being on the correct side of the line. Data points nearest to the hyperplane are considered the support vectors which can be seen in Figure 20 as data points on the dashed line. The line representing the support vector are $wx + b = 1$ and $wx + b = -1$.

Distance between the nearest data point to the hyperplanes are named as the margin. Determining the margin is necessary so that the best suitable position of the hyperplane can be found. The optimum position of the hyperplane is when it gives the largest possible margin between the data point and the hyperplane. Hard margin is used when data can be separated linearly which means the data is separated with a hyperplane without error. However, if the data is mix making it not possible to linearly separate it, a soft margin is used where the hyperplane is constructed on a position that gives the least number of misclassified data [14].

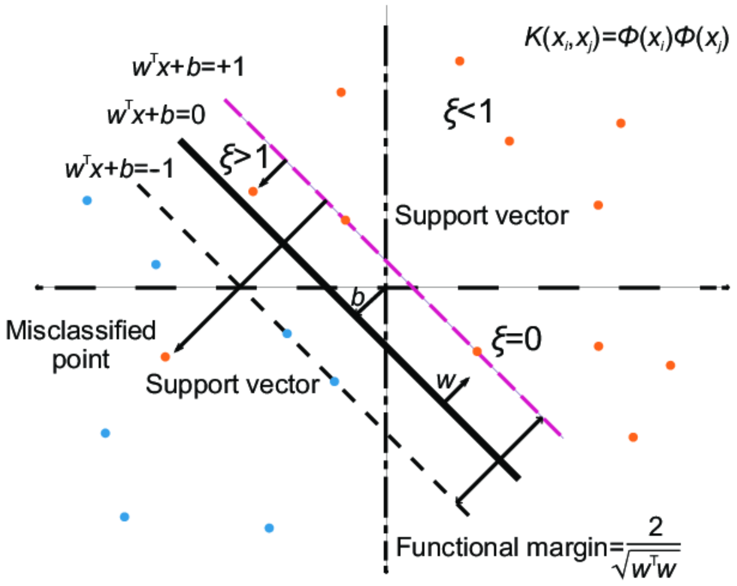


Figure 20: Concept of SVM [13]

4.5.5 Weighted K-Nearest Neighbour (WKNN)

Weighted K-nearest neighbour is a variation of the common KNN. This technique classifies data by calculating the distance of the new data entry with all the existing data as shown in Figure 21 (b). The new entry will then be classified into a group with the least distance. On top of determining the distance, WKNN also assigns weight to each existing data. Heavier weight is assigned to the data that is closer to the new entry [15]. The step procedure of WKNN for a new data entry is as shown below:

1. Determine the Euclidean distance, d, between each existing data point using the formula below:

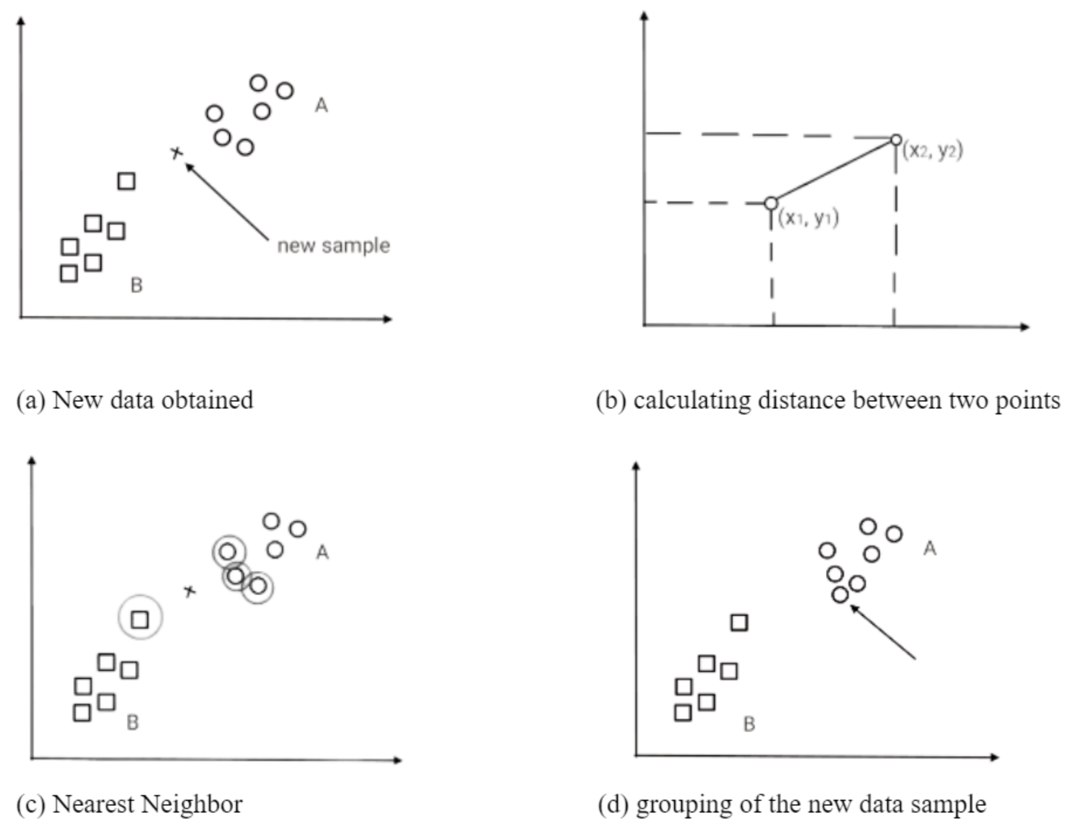
$$d = \sqrt{\sum_{k=1}^{n}(x_i - y_i)^2}$$

2. Sort the distances, d_i, from each point to the new data entry in ascending order

3. Assign weight, w_i, to the ith nearest neighbour using the formula:

$$w_i = \frac{1}{d_i^2}$$

4. Finally, the output class of the new data entry is decided by summing the weight of the same output class. The new data entry will be assigned to the group with the highest weighted sum.



(a) New data obtained (b) calculating distance between two points (c) Nearest Neighbor (d) grouping of the new data sample

Figure 21: Steps in basic KNN

4.5.6 Tri-layered Neural Network (TNN)

Tri-layered Neural Network (TNN) is a variation of Artificial Neural Network. Neural Network algorithm is heavily inspired by the human biological nervous system where a group of nodes are connected to each other like how in the human brain many of the synapses are connected between one another when making a final decision. Each input connection to a neuron is weighted and used for synaptic learning [17]. The weighted sum of the input variables transferred to a neuron is determined through an activation function as described in an equation below:

$$y = \sum w_i x_i$$

Where x_i is the input variable and w_i is the weight assigned to the input signal. The characteristic of a neuron is described by the activation function. A Neural Network is made of three important layers which are the input layer, hidden layer, and the output layer. The task of the input layer is to relocate the input vector to the hidden layer. The hidden layer can be considered as the intermediate layer linking the input to the output. Each neuron in this hidden layer is connected to all previous neuron in the input layer and the next layer of neuron as depicted in Figure 22. But neurons in the same layer are never connected to each other like how in Figure 22's h_1 neurons are not connected to each other.

The difference between a TNN and other variation of Neural Network is that TNN consist of three fully connected layers only excluding the output layer. During training, TNN utilizes backpropagation algorithm which is to update the weight and bias values by diminishing the error function in the output layer [17]. Thus, in theory it should result in a more accurate output.

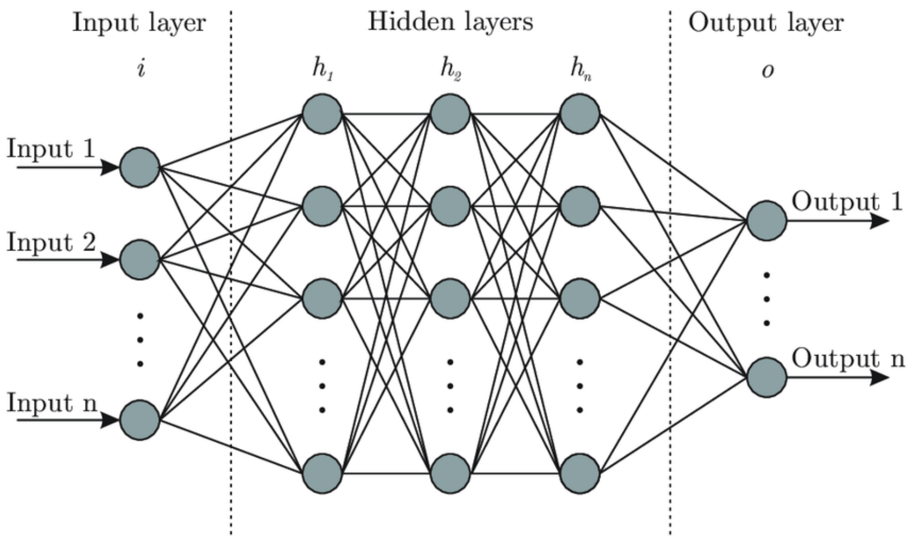


Figure 22: Basics of Neural Network [16]

4.6 Evaluation

Every classification model needs to be evaluated to determine its effectiveness. There are many statistical evaluation methods of a model available such as F1 score and Root-mean square error. In this research, the technique used to evaluate the models is confusion matrix. This method illustrates the summary of a machine learning model in predicting each class and the rate of where misclassification of a class occurs. Table 2 shows which fruit is assigned to what class.

Table 2: Assignment of fruit to a class

Class Number	Fruit
1	Unripe Banana
2	Ripe Banana
3	Overripe Banana
4	Unripe Mango
5	Ripe Mango
6	Overripe Mango
7	Unripe Tomato
8	Ripe Tomato
9	Overripe Tomato

An example shown in Figure 23, the diagonal blue boxes represent the true positive rate (TPR) which is the rate at which the predicted class by model aligns with the true class. While the false negative rate (FNR) is rate of which the model incorrectly predicts the class which is represented by orange boxes in Figure 23. By averaging TPR in the diagonal blue boxes, the overall accuracy of the model can determine.

In Figure 23, the rate at which the model correctly classify class 1 is only 37.5%. While the model misclassifies class 2, class 4, and class 7 as class 1 at the rate of 12.5%, 37.5%, and 12.5%, respectively. The model in Figure 23 has an overall training accuracy of 66.7%.

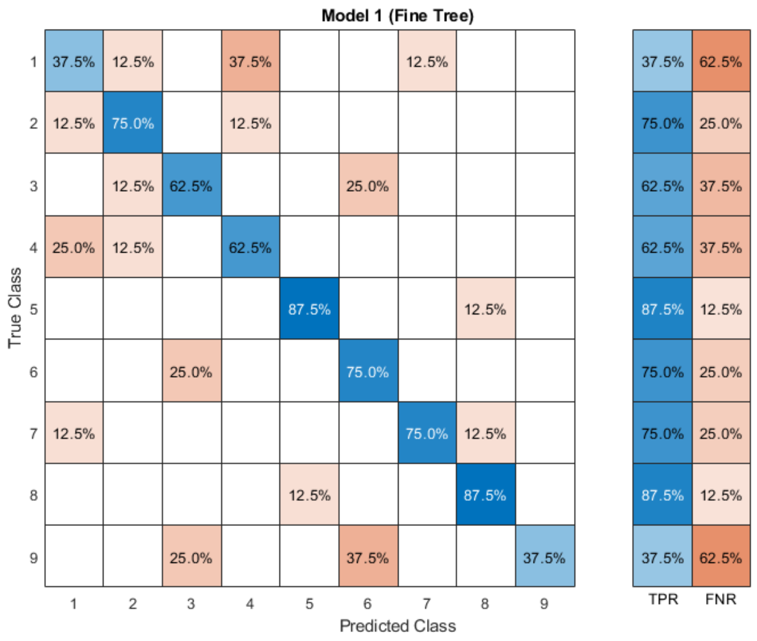


Figure 23: Training confusion matrix of a Fine tree model

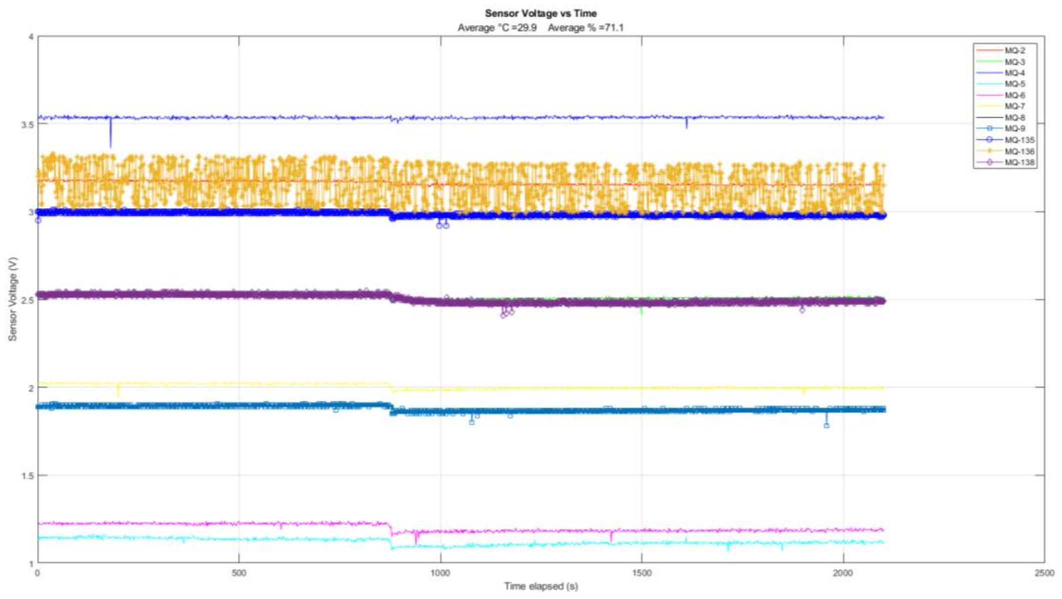
5.0 Results and Discussion

5.1 Analysing sensor response

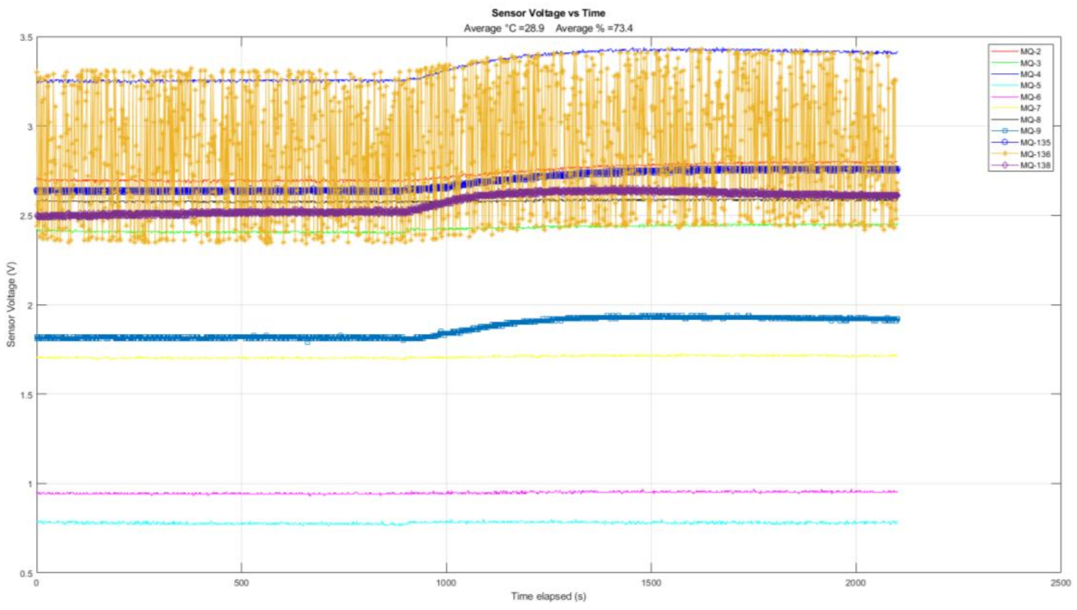
For ease of analysing the changes in sensor response for each kind of fruit at varying ripeness level, the graph to be analysed will include the response of the sensors during warm-up leading up to the moment a kind of fruit is placed inside the gas chamber which is around the 900 second's mark. As the experiment conducted to gather information on each different kind of fruit happen on different days, the baseline level of the sensor might vary due to the differences in the humidity. Unripe, ripe, and overripe fruits are the three categories into which the sensor response analysis is separated. The similarities and differences between different fruits at the same ripeness level will be explored in each group. For each group a graph representation of the raw sensor response for each fruit will be used.

5.1.1 Unripe – Bananas, Mangoes, and Tomatoes

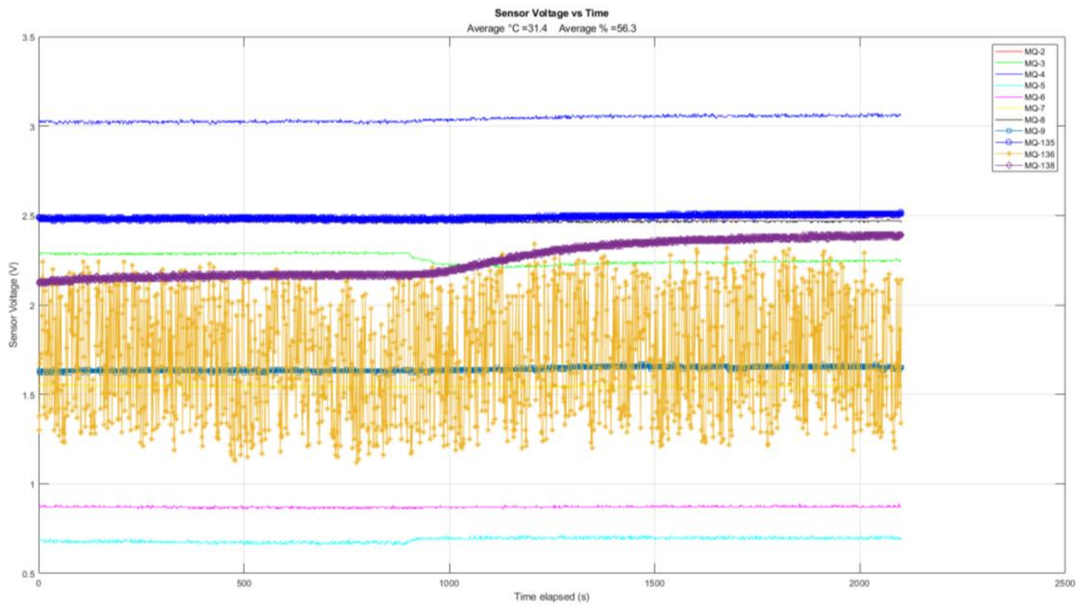
From Figure 24, there are few similarities and differences between them. The MQ-138 sensor's analogue value for both unripe mango and tomato increases until steady-state after their respective fruits are placed into the gas chamber. While for the unripe banana, the MQ-138 response decreases until a steady-state. The MQ-135 sensor response remains unchanged for both unripe banana and tomato while for unripe mango, there is a noticeable increase in the MQ-135 analogue value until steady state. For unripe mango, the analogue value for the MQ-9 increases until a steady-state after the warm-up. While there are no significant changes in the MQ-9 sensor response for unripe tomato. The MQ-4 analogue value increases until steady-state as an unripe mango is placed in the gas chamber after warm-up. While for an unripe banana and tomato, MQ-4 sensor response shows no significant change in the analogue value despite the respective fruits are placed into the chamber after warm-up. The MQ-3 sensor response decreases for both unripe banana and tomato after warm-up. Meanwhile for the unripe mango, the sensor MQ-3 response remains unchanged after the fruit is placed into the gas chamber. The MQ-2 responses in the same manner as MQ-135.



(a) Banana



(b) Mango



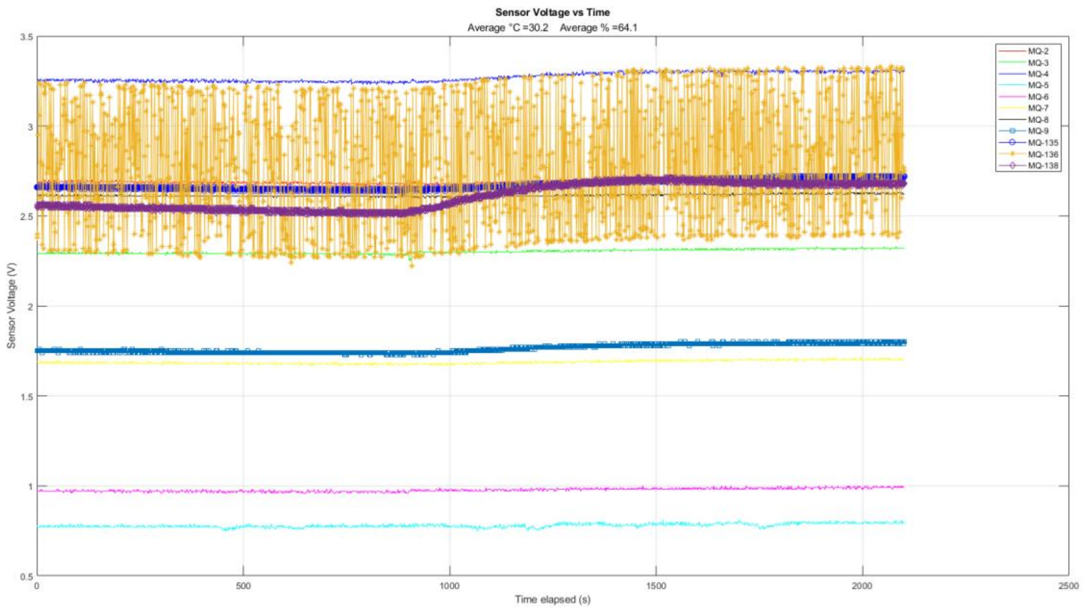
(c) Tomato

Figure 24: Different unripe fruit's sensor response

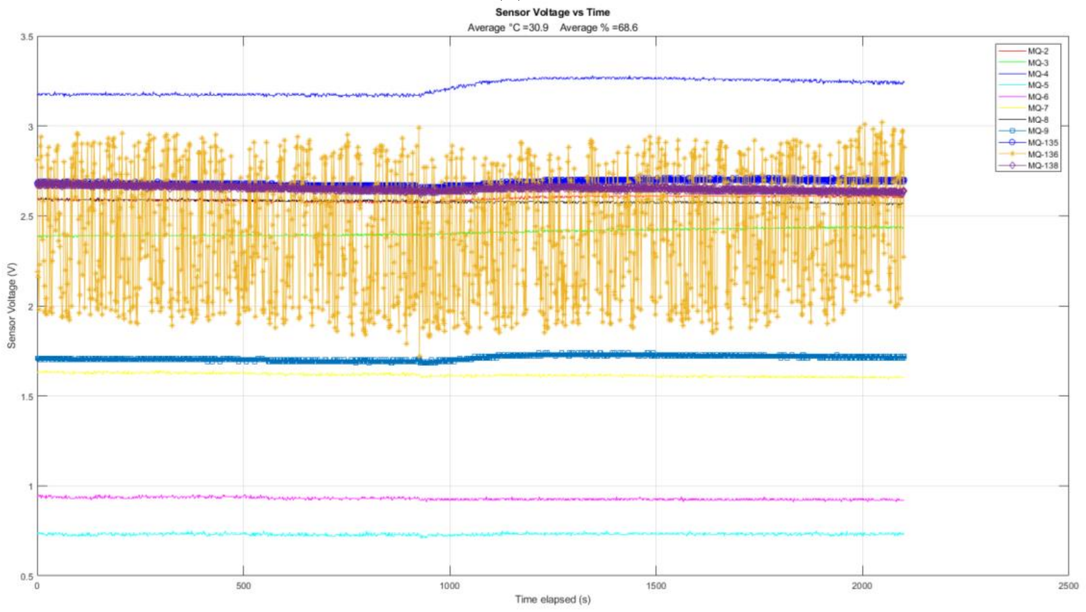
The sensors that have the common unchanged response across all three unripe fruits are the MQ-136, MQ-8, MQ-7, MQ-6, and MQ-5. Although the listed sensor's response abruptly decreases for unripe banana, the change is not significant enough to be considered hence it can be said that the sensor response remains unchanged.

5.1.2 Ripe -Bananas, Mangoes, and Tomatoes

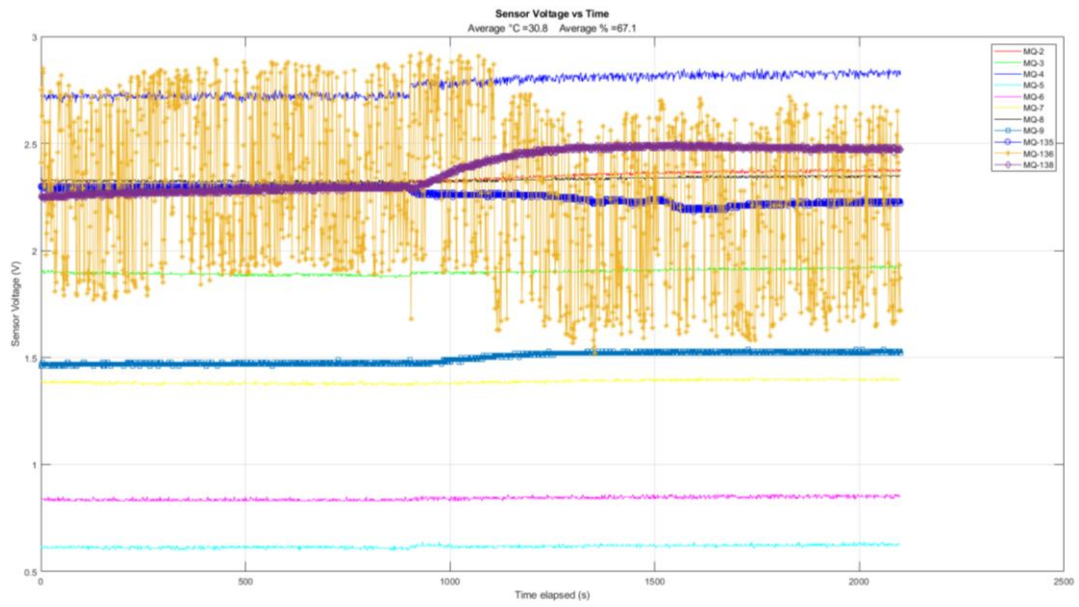
The MQ-138 sensor's analogue value increases until a steady state for both ripe banana and tomatoes while for ripe mango MQ-138 sensor response remains unchanged after warm-up. Despite the noisiness of the raw MQ-136's signal, it can be determined that for ripe banana and mango there are no significant changes in the sensor's response throughout. However, the MQ-136 has a noticeable decrease in its analogue value for ripe tomato. For both ripe banana and mango, the MQ-135 sensor response remains about constant over the course of the total 35 minutes, while for ripe tomato, the sensor first experiences an unstable decrease in its analogue value before reaching a steady-state as depicted in Figure 25(c). When a ripe mango is placed in the gas chamber after warm-up, the MQ-4 sensor's value increases until a steady state. While in the other hand, the MQ-4 analogue value abruptly increases after warm-up before remaining constant until 35 minutes for ripe tomato. The MQ-4 response was at around fixed value for ripe banana.



(a) Banana



(b) Mango



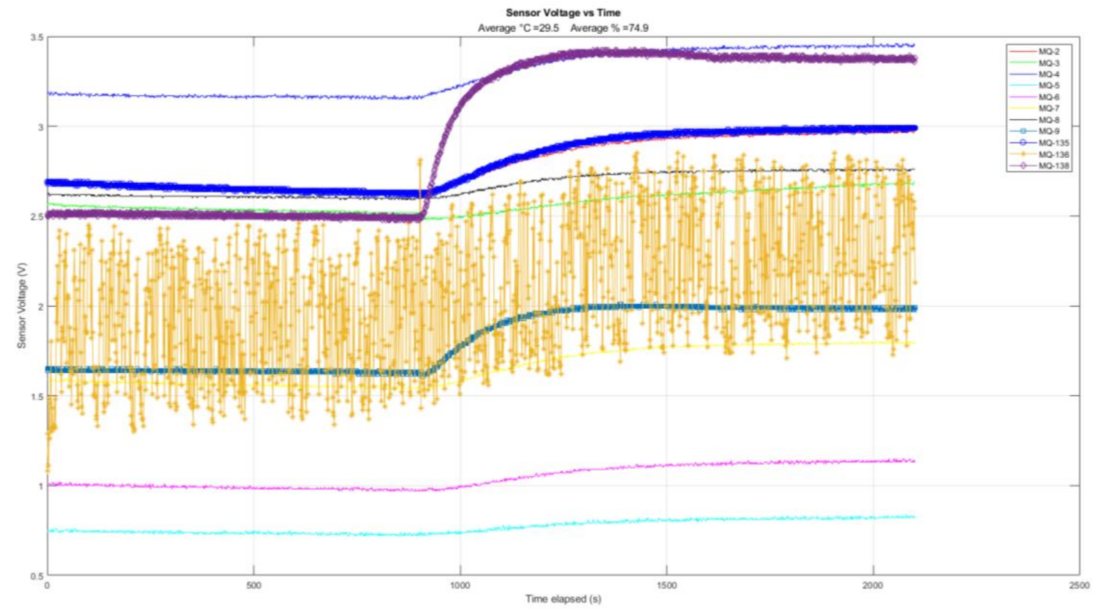
(c) Tomato

Figure 25: Different ripe fruit's sensor response

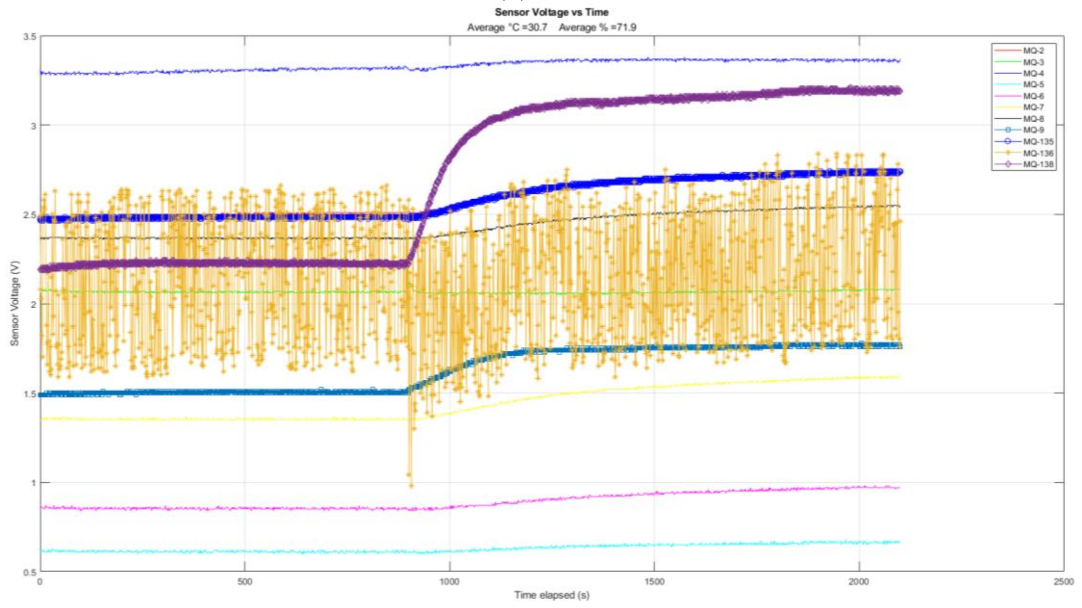
There were a few gas sensors that had around constant response throughout the 35 minutes for all ripe fruits. These sensors were the MQ-9, MQ-8, MQ-7, MQ-6, MQ-5, MQ-3, and MQ-2 as observed in Figure 25.

5.1.3 Overripe – Bananas, Mangoes, and Tomatoes

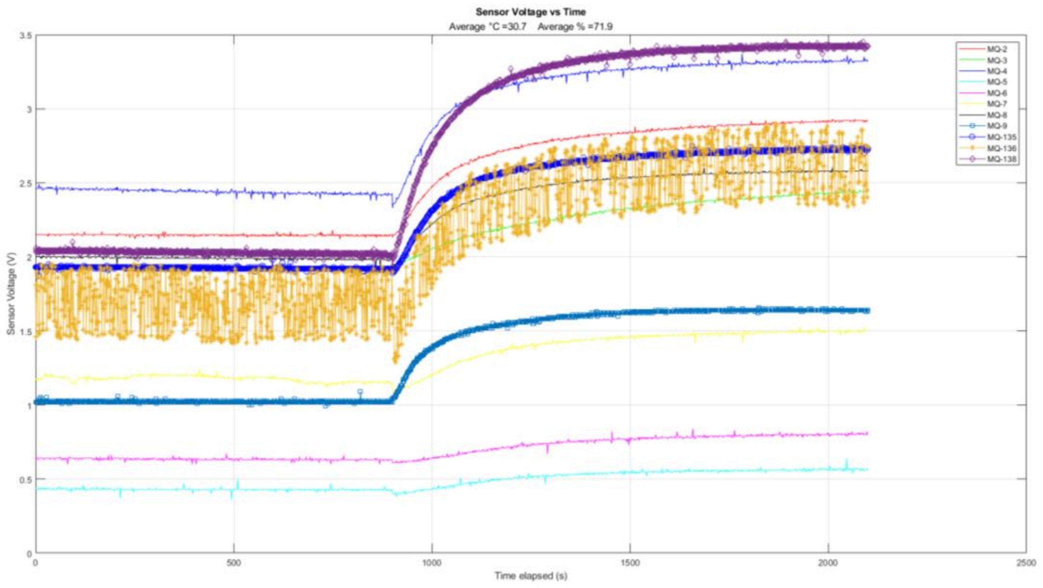
The MQ-136 sensor response increases for both overripe banana and tomato. Also, the MQ-136 increase in response for overripe tomato is much more significantly than for overripe banana. While for overripe mango, the MQ-136 sensor response had a sharp decrease the moment after warmup before returning to around initial value overtime as illustrated in Figure 26 (b). For both overripe banana and tomato, the MQ-5 sensor response increases whereas for overripe mango, the MQ-5 response had a very minor increase which can be considered an unsignificant change in the sensor response. For overripe mango, the MQ-4 sensor response remains about static throughout but for both overripe banana and tomato, the MQ-4 sensor response increases overtime with the overripe tomato having the larger increase in sensor response. The MQ-3 exhibits an increase in analogue value for both overripe banana and tomato. Even though for both overripe fruits sensor response increases, overripe tomato had a larger increase in sensor response compared to overripe banana. Meanwhile, the MQ-3 had no significant change in sensor response for overripe mango.



(a) Banana



(b) Mango



(c) Tomato

Figure 26: Different overripe fruit's sensor response

Nevertheless, a few sensors showed an increase in response across all overripe fruits. The sensors were the MQ-138, MQ-135, MQ-9, MQ-8, MQ-7, MQ-6, and MQ-2. However, these sensors showed the greatest rise in response for overripe tomato.

5.2 Feature Combination under differently processed dataset

From the data acquisition stage, a total of 90 sample data have been gathered on the fruits. Each fruit at three different ripeness level has 10 sample data. As mentioned beforehand, the training and testing data for the classification model is split 80% and 20% for each fruit at a specific ripeness level. Hence, 8 sample data were used training and the remaining 2 sample data were used for testing the model. From the feature extraction phase, different combination of features can be form as depicted in Table 3. This section will decide whether pre-processed or raw data is the optimum sort of dataset for training and testing a classification model given the collected sample data. It is also to find the best combination of features that will result in the most consistently performing machine learning model. The best model and combination feature will have above average accuracy and the least deficit in accuracy from training to testing. It is also worth noting that the training accuracy is derived entirely from the 72 sample data sets set aside for training, while the remaining 18 sample data sets are utilised to assess the trained model's testing accuracy.

Table 3: Feature Combination

Combination Number	Features	Total number of Features
1	Mean, RMS, Std, Peak to RMS, Skewness	55
2	Mean, RMS	22
3	Mean, RMS, Std	33
4	Mean, RMS, Std, Peak to RMS	44
5	RMS, Std	22
6	RMS, Std, Peak to RMS	33
7	RMS, Std, Peak to RMS, Skewness	44
8	Std, Peak to RMS	22
9	Std, Peak to RMS, Skewness	33
10	Mean, Std, Skewness	33
11	Peak to RMS, Skewness	22

5.2.1 Classification model with combination of features from raw dataset

According to Figure 27 and Table 4, the most consistent performing machine learning algorithm across all 11 possible combination of features is WKNN with an average training accuracy of 85.5%. While the inconsistent model is DT with an average training accuracy of 68.6% from the 11 possible combinations of features. It can be observed from the graph in Figure 27, WKNN with feature combination number 11 results in the model with the highest training accuracy model of 93.1%. Whereas the machine learning, TNN combined with feature combination number 9 results in the lowest possible training accuracy of 54.2%.

Table 4: Training accuracies (%) for using Raw dataset

Model	Feature Combination Number (%)										
	1	2	3	4	5	6	7	8	9	10	11
DT	80.6	63.9	76.4	72.2	73.6	73.6	69.4	62.5	58.3	68.1	55.6
LDA	62.5	87.5	90.3	73.6	90.3	87.5	77.8	83.3	79.2	90.3	72.2
GNB	87.5	81.9	88.9	86.1	88.9	86.1	84.7	59.7	70.8	84.7	68.1
LSVM	87.5	86.1	88.9	87.5	87.5	87.5	86.1	69.4	72.2	86.1	79.2
WKNN	88.9	84.7	88.9	88.9	93.1	88.9	88.9	70.8	81.9	87.5	77.8
TNN	75	75	80.6	68.1	69.4	70.8	72.2	56.9	54.2	81.9	62.5

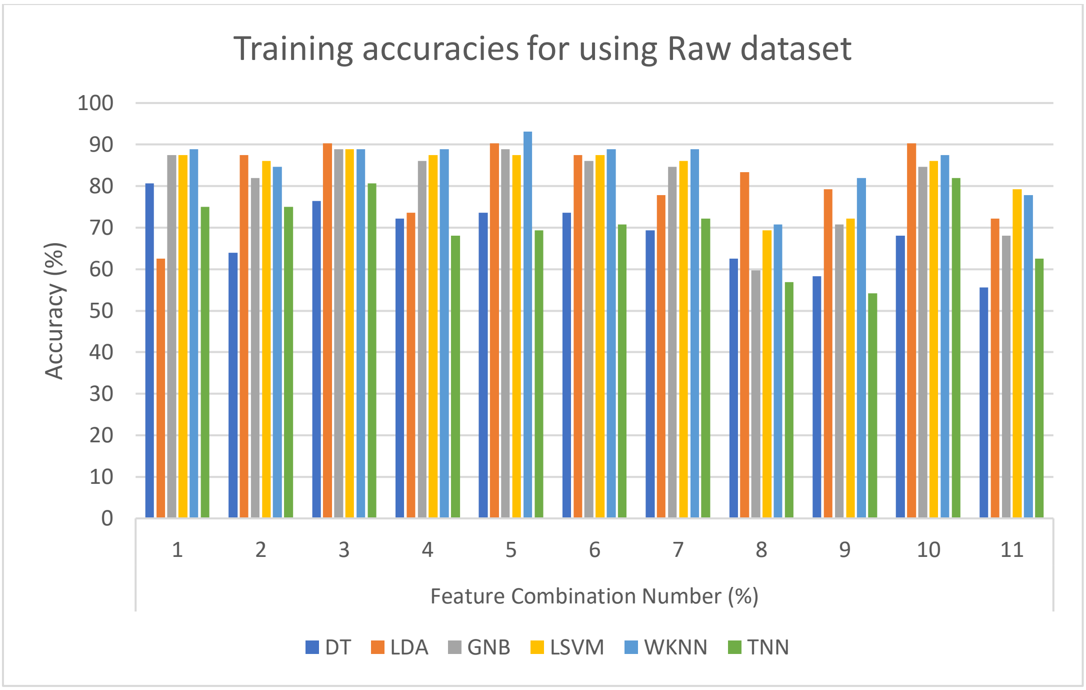


Figure 27: Training accuracies for using Raw dataset

In testing phase, the model with the highest consistent accuracy on all 11 combination of features is GNB with an average testing accuracy of 76.3% according to Figure 28 and Table 5. While on the contrary, the model with the lowest average testing accuracy from all 11 combination of features is TNN with an average testing accuracy of 55.6%. From Figure 28, it can be determined that the model with the highest testing accuracy is GNB with a testing accuracy of 83.8% when combine with feature combination number 1. Meanwhile when DT is combined with feature combination number 2, it results in the lowest possible testing accuracy of 38.9 %.

Table 5: Testing Accuracies (%) for using Raw dataset

Model	Feature Combination Number (%)										
	1	2	3	4	5	6	7	8	9	10	11
DT	61.1	38.9	61.1	61.1	61.1	61.1	61.1	50	50	61.1	50
LDA	66.7	66.7	77.8	72.2	77.8	77.8	72.2	50	44.4	77.8	55.6
GNB	83.8	61.1	83.3	77.8	83.3	77.8	83.3	72.2	77.8	83.3	55.6
LSVM	61.1	55.6	61.1	61.1	61.1	55.6	55.6	55.6	61.1	55.6	61.1
WKNN	50	72.2	61.1	66.7	61.1	66.7	55.6	66.7	66.7	50	61.1
TNN	50	50	66.7	55.6	72.2	55.6	50	55.6	50	61.1	44.4

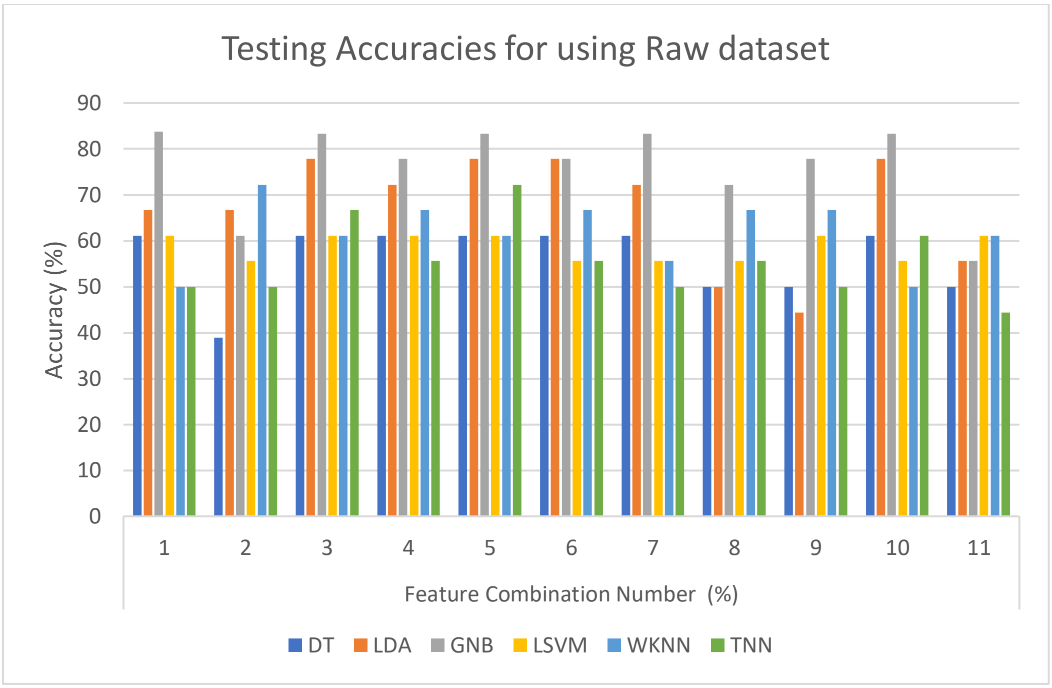


Figure 28: Testing Accuracies for using Raw dataset

From both training and testing using raw dataset, it can be concluded that the best classification model combine with a feature combination is GNB with feature combination number 1 that only has an accuracy deficit of only 3.7% from 87.5% in training to 83.8% in testing.

5.2.2 Classification model with combination of features from pre-processed dataset

From Figure 29 and Table 6, the highest consistent performing machine learning algorithm from all 11 combination of features in training is LDA with an average training accuracy of 82.3%. However, the model with the lowest average training accuracy is TNN with a mean training accuracy of 64.8% from the 11 combinations of features. As seen in Figure 29, when LDA is combined with feature combination 2, it results in the highest training accuracy with 93.1%. On the other hand, when TNN is combined with feature combination number 11 it results in the worst training accuracy of 45.8%.

Table 6: Training Accuracies (%) for using pre-processed dataset

Model	Feature Combination Number (%)										
	1	2	3	4	5	6	7	8	9	10	11
DT	72.2	72.2	63.9	70.8	75	69.4	75	61.1	61.1	66.7	47.2
LDA	66.7	93.1	90.3	76.4	95.8	91.7	77.8	80.6	76.4	88.9	68.1
GNB	84.7	80.6	86.1	87.5	86.1	84.7	87.5	56.9	68.1	84.7	65.3
LSVM	86.1	84.7	86.1	87.5	90.3	87.5	83.3	66.7	68.1	88.9	63.9
WKNN	81.9	83.3	91.7	84.7	90.3	84.7	83.3	62.5	55.6	76.4	55.6
TNN	68.1	79.2	75	75	72.2	72.2	66.7	47.2	47.2	63.9	45.8

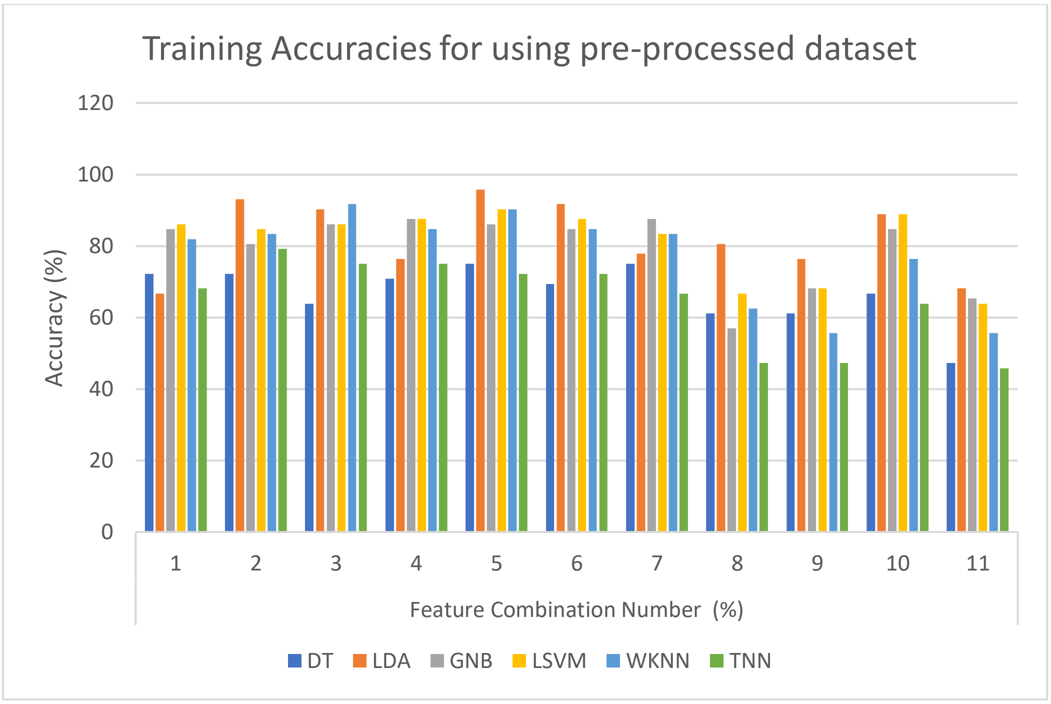


Figure 29: Training Accuracies for using pre-processed dataset

From Figure 30 and Table 7, in the testing, the model with the best average testing accuracy from all 11 combination of features is WKNN with an average testing accuracy of 49.5%. While GNB on the other hand, has the worst average testing accuracy from all 11 combination of features with 31.8%. From Figure 30, it can be determined that the highest testing accuracy can be achieved when WKNN is combined with feature combination number 2 with a testing accuracy of 72.2%. Whereas when TNN is combined with feature combination 10 it results in the lowest possible testing accuracy of 16.7%.

Table 7: Testing Accuracies (%) for using pre-processed dataset

Model	Feature Combination Number (%)										
	1	2	3	4	5	6	7	8	9	10	11
DT	55.6	38.9	55.6	55.6	50	50	50	27.8	33.3	55.6	22.2
LDA	38.9	27.8	27.8	27.8	66.7	66.7	44.4	22.2	33.3	61.1	22.2
GNB	27.8	61.1	44.4	27.8	38.9	22.2	22.2	22.2	22.2	38.9	22.2
LSVM	50	55.6	55.6	55.6	55.6	44.4	44.4	22.2	27.8	66.7	33.3
WKNN	44.4	72.2	55.6	50	61.1	44.4	44.4	44.4	38.9	44.4	44.4
TNN	38.9	50	55.6	27.8	33.3	16.7	44.4	22.2	33.3	16.7	44.4

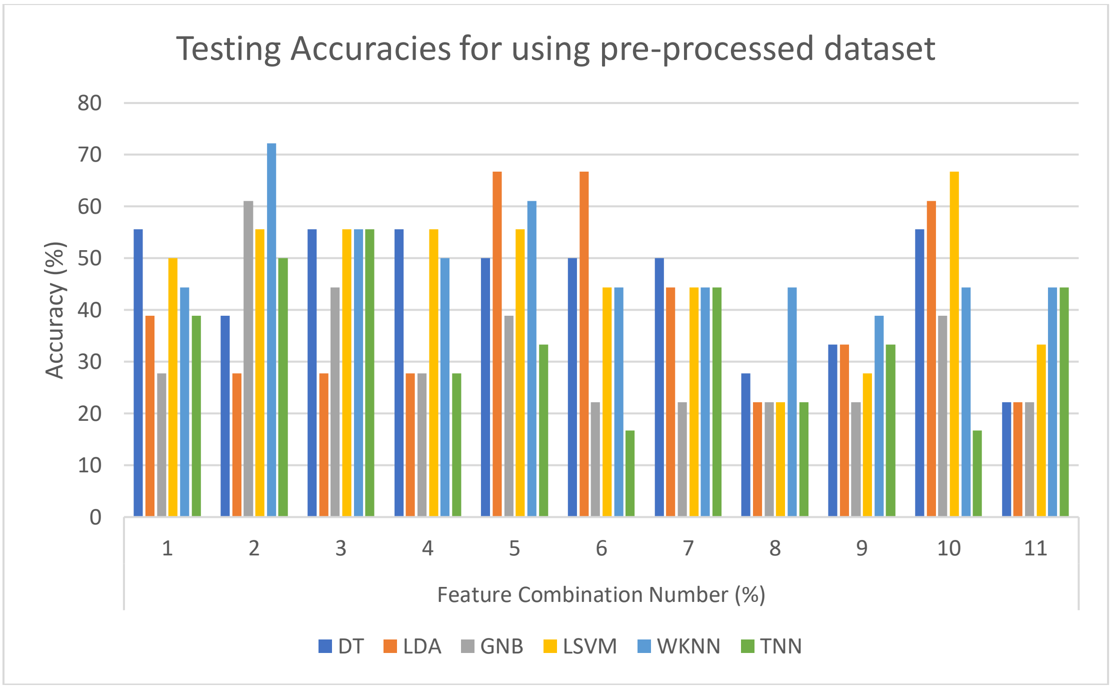


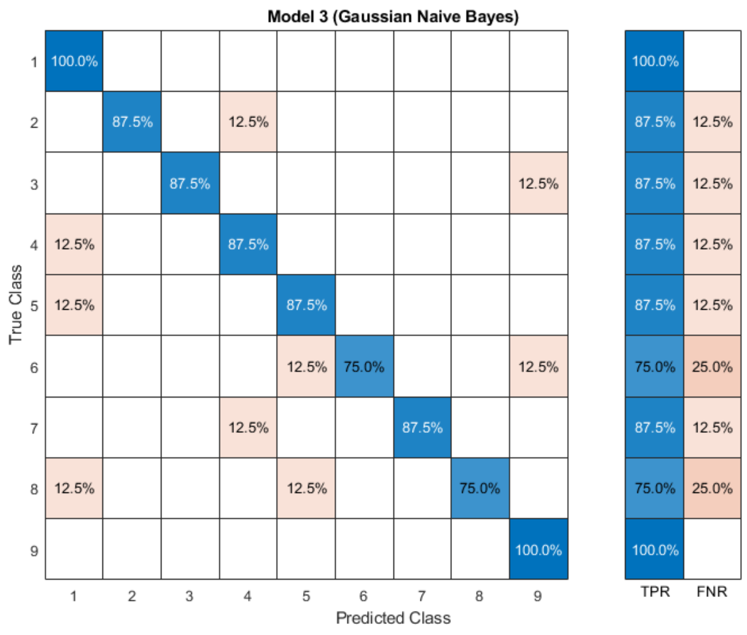
Figure 30: Testing Accuracies for using pre-processed dataset

From the training and testing using pre-processed data, it can be determined that the best machine learning algorithm is WKNN when combined with feature combination number 2 as it only results in a deficit of 11.1% from 83.3% in training to 72.2% testing and has an above average accuracy.

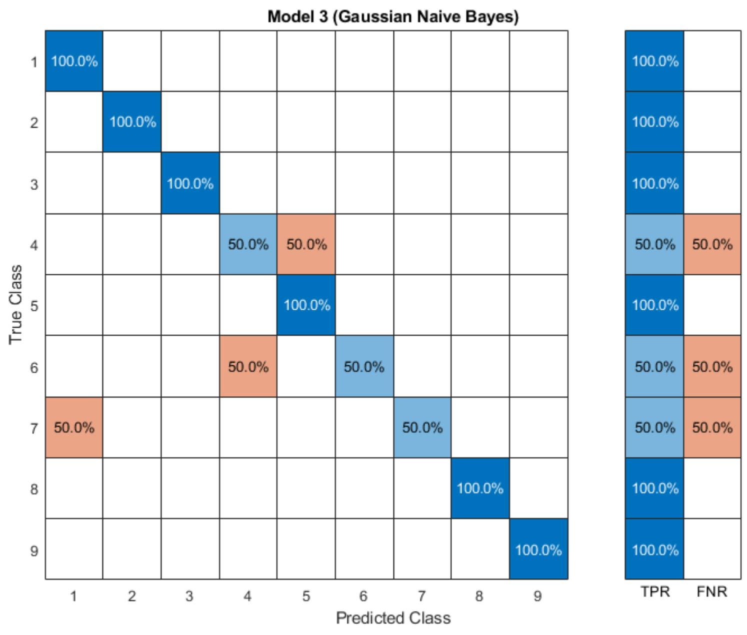
5.2.3 Final decision on classification model

From the previous two sub-sections, it can be concluded that using raw dataset can give the more consistent classification model. This is when using raw dataset, GNB combined with feature combination number 1 only has a deficit of 3.7% in accuracy from training to testing compared to when using a pre-processed dataset, WKNN combined with feature combination number 2, results in a larger deficit of 11.1% in accuracy from training to testing.

Figure 31 showcases the confusion matrix for the best classification model which GNB with feature combination number 1. From Figure 31 (a), in training this classification model often misclassify class 6 and class 8 as both had a FNR of 25%. While having a TPR of 100% for both class 1 and class 9. Meanwhile in testing as shown in Figure 31 (b), the highest FNR is 50% which happens at class 4, class 6, and class 7. Despite this, in other classes it has a 100% TPR.



(a) Training confusion matrix



(b) Testing confusion matrix

Figure 31: Best model’s confusion matrix

6.0 Conclusion and Future recommendations

According to the analysis of graphs representing each fruit at three distinct ripeness levels, there are some similarities and variances in sensor response for each fruit in the same ripeness group. The differences were utilized by the classification model when learning their respective datasets. Furthermore, the best possible classification model given the gathered sample data is Gaussian Naïve Bayes when combined with feature combination number 1 as it resulted in accuracy of 87.5% and 83.8% for training and testing, respectively. Making this a very consistent model as it only has a loss of 3.7% from training to testing phase. The input dataset for this model were a combination of features that were extracted from raw sample data gathered during data acquisition stage.

Even though in theory, the usage of signal pre-processing was supposed to improve classification accuracy as the signals are more refined than the raw signals obtained from data gathering, but in this research, it has proven to be otherwise. Reasons for this could be that the refining might altered too much of the original signal that it decreases accuracy in classification as most signals would look like each other. Besides that, one of the main challenges faced with the making of the gas chamber for the electronic nose is the power consumption of the gas sensors, as the MQ-series gas sensors uses a lot of power [1]. The 3.2 V supplied to each gas sensors might not be enough as it can be observed from Figure 24, 25, and 26, the analogue signals produced by each sensor is noticeably noisy.

Future improvements that could be made to this electronic system could be to include other sensors such as colour sensor. As generally, people rely on colour of the fruit also to determine the ripeness of it.

7.0 References

[1] Wilson, A. D., & Baietto, M. (2009). Applications and advances in electronic-nose technologies. *Sensors, 9*(7), 5099-5148.

[2] A. P. D. Heredia, F. R. Cruz, J. R. Balbin and W. -Y. Chung, "Olfactory classification using electronic nose system via artificial neural network," 2016 IEEE Region 10 Conference (TENCON), 2016, pp. 3569-3574, doi: 10.1109/TENCON.2016.7848722.

[3] Dautov, Ç. P., & Özerdem, M. S. (2018, May). Wavelet transform and signal denoising using Wavelet method. In *2018 26th Signal Processing and Communications Applications Conference (SIU)* (pp. 1-4). Ieee.

[4] Johnstone, I. M., and B. W. Silverman. “Needles and Straw in Haystacks: Empirical Bayes Estimates of Possibly Sparse Sequences.” Annals of Statistics, Vol. 32, Number 4, pp. 1594–1649, 2004.

[5] Pearson, R. K., Neuvo, Y., Astola, J., & Gabbouj, M. (2016). Generalized hampel filters. EURASIP Journal on Advances in Signal Processing, 2016(1), 1-18.

[6] Smith, S. W. (1997). The Scientist and Engineer's Guide to. Digital Signal Processing,. California Technical Publishing, San, Diego, CA.

[7] E. Papageorgiou, C. Stylios and P. Groumpos, "A Combined Fuzzy Cognitive Map and Decision Trees Model for Medical Decision Making," 2006 International Conference of the IEEE Engineering in Medicine and Biology Society, 2006, pp. 6117-6120, doi: 10.1109/IEMBS.2006.260354.

[8] Data mining EEG signals in depression for their diagnostic value - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Linear-discriminant-analysis_fig5_288002528 [accessed 14 Sep, 2022]

[9] Subasi, A., & Gursoy, M. I. (2010). EEG signal classification using PCA, ICA, LDA and support vector machines. Expert systems with applications, 37(12), 8659-8666.

[10] Priyankur Sarkar. (2019, September 30). What is linear discriminant analysis(lda)? What is Linear Discriminant Analysis(LDA)? Retrieved September 10, 2022, from https://www.knowledgehut.com/blog/data-science/linear-discriminant-analysis-for-machine-learning

[11] Smoothness without Smoothing: Why Gaussian Naive Bayes Is Not Naive for Multi-Subject Searchlight Studies - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Illustration-of-how-a-Gaussian-Naive-Bayes-GNB-classifier-works-For-each-data-point_fig8_255695722 [accessed 14 Sep, 2022]

[12] Griffis, J. C., Allendorfer, J. B., & Szaflarski, J. P. (2016). Voxel-based Gaussian naïve Bayes classification of ischemic stroke lesions in individual T1-weighted MRI scans. Journal of neuroscience methods, 257, 97-108.

[13] EasySVM: A visual analysis approach for open-box support vector machines - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/A-linear-support-vector-machine_fig1_315316855 [accessed 14 Sep, 2022]

[14] S. Ghosh, A. Dasgupta and A. Swetapadma, "A Study on Support Vector Machine based Linear and Non-Linear Pattern Classification," 2019 International Conference on Intelligent Sustainable Systems (ICISS), 2019, pp. 24-28, doi: 10.1109/ISS1.2019.8908018.

[15] H. Yigit, "A weighting approach for KNN classifier," 2013 International Conference on Electronics, Computer and Computation (ICECCO), 2013, pp. 228-231, doi: 10.1109/ICECCO.2013.6718270.

[16] Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Artificial-neural-network-architecture-ANN-i-h-1-h-2-h-n-o_fig1_321259051 [accessed 14 Sep, 2022]

[17] Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Artificial-neural-network-architecture-ANN-i-h-1-h-2-h-n-o_fig1_321259051 [accessed 14 Sep, 2022]

[18] Occupancy Estimation and People Flow Prediction in Smart Environments - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Structure-of-a-simple-decision-tree_fig2_322273002 [accessed 18 Sep, 2022]

8.0 Appendix

MATLAB code used for Data acquisition

```
%clear previous session's work
clear
clc

%arduinosetup(); %to setup the board again

%-------------------------------------------------Setup function
N = 1; %data collector counter
Maxdata = 300;  % pre-allocate memory
MaxTime = 1200; % 20 minutes
warmup_time = 900; % 15 minutes
prep_fruit = 15; % 15 seconds for fruit to prep into chamber

%Directory of Excel folder
Excel = 'C:\Users\Arfan Danial\OneDrive - University of Nottingham Malaysia\Summer 
2022\Summer Research Intern Essentials\Gathered Data in Excel\';
ExcelType = '.xlsx';

%Directory of Plot folder
Plot = 'C:\Users\Arfan Danial\OneDrive - University of Nottingham Malaysia\Summer 
2022\Summer Research Intern Essentials\Plotted graphs\';
PlotType = '.png';

%sum for humidity and temperature
%to find average for each testing
humidity = zeros(Maxdata,1,'double');
temperature = zeros(Maxdata,1,'double');

%initializing arduino board object
a = arduino("COM7","Mega2560","Libraries","Adafruit/DHTxx");

%initialize the DHT22 sensor object
sensor = addon(a, 'Adafruit/DHTxx', 'D7','DHT22');

%store time
time = 0;
%array to store timer
stoptime = zeros(Maxdata,1,'double');

%store each mq sensor values in its own array
%use array of zeroes to preallocate memory
a2 = zeros(Maxdata,1,'double');
a3 = zeros(Maxdata,1,'double');
a4 = zeros(Maxdata,1,'double');
a5 = zeros(Maxdata,1,'double');
a6 = zeros(Maxdata,1,'double');
a7 = zeros(Maxdata,1,'double');
a8 = zeros(Maxdata,1,'double');
a9 = zeros(Maxdata,1,'double');
a135 = zeros(Maxdata,1,'double');
a136 = zeros(Maxdata,1,'double');
a138 = zeros(Maxdata,1,'double');

%-------------------------------------------------main function

%Ask for user input
```

```
filename1 = input('Excel Filename: ','s'); %name of excel file
sheet = input('Sheet No: '); %specify what sheet in a Excel File
filename2 = input('Graph Name: ','s'); %name of plot picture

%asking if user wants to warm-up the sensors
fprintf('\nDo you want to warm-up? 1:Yes | 2:No')
warmup = input('\n');

%create full directory
Exceldirectory = strcat(Excel,filename1,ExcelType);
Plotdirectory = strcat(Plot,filename2,PlotType);

%--------------------------------------------------loop function
while (1)
    if (time<MaxTime)
        %read DHT22 sensor
        temperature(N) = readTemperature(sensor);
        humidity(N) = readHumidity(sensor);

        tic %start timer

        %reading sensor voltage of each sensor
        %set to 2 decimal places for each sensor voltage%.2f
        mq2 = round(a.readVoltage('A10'),2);
        mq3 = round(a.readVoltage('A8'),2);
        mq4 = round(a.readVoltage('A6'),2);
        mq5 = round(a.readVoltage('A5'),2);
        mq6 = round(a.readVoltage('A2'),2);
        mq7 = round(a.readVoltage('A1'),2);
        mq8 = round(a.readVoltage('A0'),2);
        mq9 = round(a.readVoltage('A9'),2);
        mq135 = round(a.readVoltage('A7'),2);
        mq136 = round(a.readVoltage('A4'),2);
        mq138 = round(a.readVoltage('A3'),2);

        pause(1);%delay 1 seconds before next step
        currentTime = toc; %stop timer and record

        time = time + round(currentTime,2);% timer between sensor input

        %if statement to see if user wanted to warm-up sensors
        %and if the time is more than warm-up time or not
        if(warmup == 1 && (time <= warmup_time))
            %printing time before warmup ends
            fprintf('Sensor has warmed up for %.2fs\n',time);
        else
            %purpose of if statement is to set the first set of inputs at 0
seconds
            %due to delay start of around 2 seconds everytime first time run
            if(N == 1)
                fprintf('\nExperiment will start in %d seconds!!\n',prep_fruit);
                pause(prep_fruit);
                warmup = 2; %change so that the program knows it doesnt need to
warm-up anymore
                time = 0; %reset timer for actual time in the experiment
                stoptime(N) = 0.00;
            else
                %put the time into an array
                stoptime(N) = time;
            end

            %replace the zeros in each index in an array
            a2(N) = mq2;
            a3(N) = mq3;
            a4(N) = mq4;
            a5(N) = mq5;
            a6(N) = mq6;
            a7(N) = mq7;
            a8(N) = mq8;
            a9(N) = mq9;
            a135(N) = mq135;
            a136(N) = mq136;
            a138(N) = mq138;

            %print values onto command window

printvalue(N,time,temperature(N),humidity(N),mq2,mq3,mq4,mq5,mq6,mq7,mq8,mq9,mq135
,mq136,mq138);

            N = N+1; %increment sample data collected
        end

    else
        N = N-1; %minus 1 due to bug in indexing for total collected data per
sensor
        break
    end

end

%find average humidity and temperature for each testing and print end
%message
[temp, humi] = weather_summary(N,temperature,humidity);

%call function to export raw data into Excel
toExcel(Exceldirectory,sheet,stoptime,a2,a3,a4,a5,a6,a7,a8,a9,a135,a136,a138);

%call function to make the graph of sensor voltage vs time
graphing(Plotdirectory,stoptime,a2,a3,a4,a5,a6,a7,a8,a9,a135,a136,a138,temp,humi);

%call function to make subplots of each sensor into a singular folder
subplots(filename2,PlotType,stoptime,a2,a3,a4,a5,a6,a7,a8,a9,a135,a136,a138);

%clear the board object created
clear sensor
clear a


%----------------------------Function definition after program

function
printvalue(N,time,temperature,humidity,mq2,mq3,mq4,mq5,mq6,mq7,mq8,mq9,mq135,mq136
,mq138)
    %printing values of sensor on command window
    fprintf("Data: %d\n",N);
    fprintf("Time: %.2f s\n",time);
    fprintf('Temperature: %.2f °C\n', temperature);
    fprintf('Humidity: %.2f %% \n', humidity);
    fprintf("MQ-2: %.2f\n", mq2);
    fprintf("MQ-3: %.2f\n", mq3);
    fprintf("MQ-4: %.2f\n", mq4);
    fprintf("MQ-5: %.2f\n", mq5);
    fprintf("MQ-6: %.2f\n", mq6);
    fprintf("MQ-7: %.2f\n", mq7);
    fprintf("MQ-8: %.2f\n", mq8);
    fprintf("MQ-9: %.2f\n", mq9);
    fprintf("MQ-135: %.2f\n", mq135);
    fprintf("MQ-136: %.2f\n", mq136);
    fprintf("MQ-138: %.2f\n", mq138);
    fprintf("\n");
end

function [temp, humi] = weather_summary(N,temperature,humidity)
    %remove 'Nan' values from array
    temperature = rmmissing(temperature);
    humidity = rmmissing(humidity);

    %calculate average over N number of data
    temp = round(mean(temperature),1);
    humi = round(mean(humidity),1);

    %print end messages
    fprintf('------------------------------------------\n');
    fprintf('Program Summary\n');
    fprintf('------------------------------------------\n');
    fprintf('Data collected: %d\n',N);
    fprintf("Average Temperature: %.2f °C\n",temp);
    fprintf("Average Humidity: %.2f %%\n",humi);

end

function
toExcel(Exceldirectory,sheet,stoptime,a2,a3,a4,a5,a6,a7,a8,a9,a135,a136,a138)
    %array of headers to be inputed in Excel file
    header = ["Time(s)","MQ-2(V)","MQ-3(V)","MQ-4(V)","MQ-5(V)","MQ-6(V)","MQ-
7(V)","MQ-8(V)","MQ-9(V)","MQ-135(V)","MQ-136(V)","MQ-138(V)"];

    %write the header in the Excel file
    writematrix(header, Exceldirectory,"Sheet",sheet);

    %write time and sensors voltage in the Excel file
    %different sensor is input in different colomn
    %written on the 'sheet' of the Excel file
    writematrix(stoptime,Exceldirectory,'Sheet',sheet,'Range','A2');
    writematrix(a2,Exceldirectory,'Sheet',sheet,'Range','B2');
    writematrix(a3,Exceldirectory,'Sheet',sheet,'Range','C2');
    writematrix(a4,Exceldirectory,'Sheet',sheet,'Range','D2');
    writematrix(a5,Exceldirectory,'Sheet',sheet,'Range','E2');
    writematrix(a6,Exceldirectory,'Sheet',sheet,'Range','F2');
    writematrix(a7,Exceldirectory,'Sheet',sheet,'Range','G2');
    writematrix(a8,Exceldirectory,'Sheet',sheet,'Range','H2');
    writematrix(a9,Exceldirectory,'Sheet',sheet,'Range','I2');
    writematrix(a135,Exceldirectory,'Sheet',sheet,'Range','J2');
    writematrix(a136,Exceldirectory,'Sheet',sheet,'Range','K2');
    writematrix(a138,Exceldirectory,'Sheet',sheet,'Range','L2');
end
```

```
function
graphing(Plotdirectory,stoptime,a2,a3,a4,a5,a6,a7,a8,a9,a135,a136,a138,temp,humi)
    %convert humidity and temperature to a string for printing
    temp = string(temp);
    humi = string(humi);
    %create the subtitle for humidity and temperature
    sub = strcat('Average °C = ',temp,'    Average % = ',humi);

    %plot the graph
    plot(stoptime,a2,'-r');
    hold on %used to plot more than one graph
    plot(stoptime,a3,'-g');
    plot(stoptime,a4,'-b');
    plot(stoptime,a5,'-c');
    plot(stoptime,a6,'-m');
    plot(stoptime,a7,'-y');
    plot(stoptime,a8,'-k');
    plot(stoptime,a9,'-s');
    plot(stoptime,a135,'b-o');
    plot(stoptime,a136,'-*');
    plot(stoptime,a138,'-d');
    hold off %after finishing the plotting

    %necessary labelling of the graph
    grid on;
    title('Sensor Voltage vs Time');
    subtitle(sub);
    xlabel('Time elapsed (s)'); %x-labelling
    ylabel('Sensor Voltage (V)'); %y-labelling
    legend('MQ-2','MQ-3','MQ-4','MQ-5','MQ-6','MQ-7','MQ-8','MQ-9','MQ-135','MQ-
136','MQ-138');

    %exporting plotted graph as png
    f = gcf; %to get the current figure and assign to 'f'
    exportgraphics(f,Plotdirectory); %export the plot
end

function
subplots(filename2,PlotType,stoptime,a2,a3,a4,a5,a6,a7,a8,a9,a135,a136,a138)
    %setting to correct directory and making a new folder
    subplot_directory = strcat('C:\Users\Arfan Danial\OneDrive - University of
Nottingham Malaysia\Summer 2022\Summer Research Intern Essentials\Subplot
graphs\',filename2);
    mkdir(subplot_directory);

    plot(stoptime,a2,'-r');
    f = gcf;
    exportgraphics(f,strcat(subplot_directory,'\MQ-2',PlotType));

    plot(stoptime,a3,'-g');
    exportgraphics(f,strcat(subplot_directory,'\MQ-3',PlotType));

    plot(stoptime,a4,'-b');
    exportgraphics(f,strcat(subplot_directory,'\MQ-4',PlotType));

    plot(stoptime,a5,'-c');
    exportgraphics(f,strcat(subplot_directory,'\MQ-5',PlotType));

    plot(stoptime,a6,'-m');
    exportgraphics(f,strcat(subplot_directory,'\MQ-6',PlotType));

    plot(stoptime,a7,'-y');
    exportgraphics(f,strcat(subplot_directory,'\MQ-7',PlotType));

    plot(stoptime,a8,'-k');
    exportgraphics(f,strcat(subplot_directory,'\MQ-8',PlotType));

    plot(stoptime,a9,'-s');
    exportgraphics(f,strcat(subplot_directory,'\MQ-9',PlotType));

    plot(stoptime,a135,'b-o');
    exportgraphics(f,strcat(subplot_directory,'\MQ-135',PlotType));

    plot(stoptime,a136,'-*');
    exportgraphics(f,strcat(subplot_directory,'\MQ-136',PlotType));

    plot(stoptime,a138,'-d');
    exportgraphics(f,strcat(subplot_directory,'\MQ-138',PlotType));
end
```

MATLAB code used for preparing dataset

```
retry = 'y';

while retry == 'y'
    %clear previous session's memory
    clear
    clc

    %set max number of samples to extract from file
    maxrows = 1:840;

    training_model = 'C:\Users\Arfan Danial\OneDrive - University of Nottingham 
Malaysia\Summer 2022\Summer Research Intern Essentials\Datasets for Model\Training 
Dataset';
    testing_model = 'C:\Users\Arfan Danial\OneDrive - University of Nottingham 
Malaysia\Summer 2022\Summer Research Intern Essentials\Datasets for Model\Testing 
Dataset';
    graph_directory1 = 'C:\Users\Arfan Danial\OneDrive - University of Nottingham 
Malaysia\Summer 2022\Summer Research Intern Essentials\Datasets for Model\Plot 
Figures\Raw Figures\';
    graph_directory2 = 'C:\Users\Arfan Danial\OneDrive - University of Nottingham 
Malaysia\Summer 2022\Summer Research Intern Essentials\Datasets for Model\Plot 
Figures\Preprocessed Figures\';

    %creating common directory where raw data is stored
    directory = 'C:\Users\Arfan Danial\OneDrive - University of Nottingham 
Malaysia\Summer 2022\Summer Research Intern Essentials\Gathered Data in Excel\';
    fileType = '.xlsx';

    %Ask user if want to prepare training or testing dataset
    decision = input('Dataset Preparation? (1:Training | 2:Testing) ');
    choice = input('1:Raw data or 2:Preprocessed data |  ');
    filename = input('Filename: ','s');

    %N determines number of sheets to be read
    N = input('Number of Sheet: ');
    %call function to determine the fruit and its ripeness based on
    %filename
    fruit_type = determineFruit(filename);

    for loop = 1:1:N

        %ask user for specific sheet to be read from the Excel file
        sheet_number = input('\nSheet Number: ');
        Openfile = readmatrix(strcat(directory, filename, fileType), "Sheet",
sheet_number);

        %organize the read file into different sensor data
        %reduce the noise of the signal of each sensor response
        Time = Openfile(maxrows,1);
        MQ2 = Openfile(maxrows,2);
        MQ3 = Openfile(maxrows,3);
        MQ4 = Openfile(maxrows,4);
        MQ5 = Openfile(maxrows,5);
        MQ6 = Openfile(maxrows,6);
        MQ7 = Openfile(maxrows,7);
        MQ8 = Openfile(maxrows,8);
        MQ9 = Openfile(maxrows,9);
        MQ135 = Openfile(maxrows,10);
        MQ136 = Openfile(maxrows,11);
        MQ138 = Openfile(maxrows,12);


        %create matrix of all 11 sensor responses
        data_table = [MQ2,MQ3,MQ4,MQ5,MQ6,MQ7,MQ8,MQ9,MQ135,MQ136,MQ138];

        %check if user wants to do preprocessing on signals
        %raw data
        if (decision == 1 && choice == 1)
            %create directory for saving figures into raw data figures file
            Plotdirectory =
strcat(graph_directory1,string(fruit_type),'.',string(sheet_number),".png");

            %plot the raw data graph and save it
            graphing(Plotdirectory,data_table);

        %preprocessed data
        elseif (decision == 1 && choice == 2)
            %perform signal pre-processing on all 11 sensor responses
            data_table = preprocess(data_table);

            %create directory for saving figures into preprocesed data figures
file
            Plotdirectory =
strcat(graph_directory2,string(fruit_type),'.',string(sheet_number),".png");

            %plot the preprocesed data graph and save it
            graphing(Plotdirectory,data_table);
        end

        %extract features from each sensor response
        %round the feature's value to 3 decimal places
        feature_extracted = round(featureExtract(data_table),3);

        %assign appropiate target
```

```matlab
        target = fruit_type;

        %create array that combines features and classifier
        final_array = [feature_extracted,target];

        %export data to a specified spreadsheet
        %readExport(decision, choice, training_model, testing_model, final_array);

    end

    %ask if user want to do another Excel file
    retry = input("\nDo you want to continue? ",'s');
end


%function definition--------------------------
function fruit = determineFruit(fruit_type)
    if (fruit_type == "CA")
        fruit = 0;

    elseif(fruit_type == "CURB")
        fruit = 1;

    elseif (fruit_type == "CRB")
        fruit = 2;

    elseif (fruit_type == "CORB")
        fruit = 3;

    elseif (fruit_type == "CURM")
        fruit = 4;

    elseif (fruit_type == "CRM")
        fruit = 5;

    elseif (fruit_type == "CORM")
        fruit = 6;

    elseif (fruit_type == "CURT")
        fruit = 7;

    elseif (fruit_type == "CRT")
        fruit = 8;

    elseif (fruit_type == "CORT")
        fruit = 9;

    else
        fruit = 10;

    end

end

function postprocessed = preprocess(preprocessed)
    %step 1: wavelet signal denoising
    processed_1 = wdenoise(preprocessed, 9, ...
                    'Wavelet', 'sym4', ...
                    'DenoisingMethod', 'Bayes', ...
                    'ThresholdRule', 'Median', ...
                    'NoiseEstimate', 'LevelIndependent');

    %step 2: Outlier Removal using Hampel Filter
    processed_2 = hampel(processed_1);

    %step 3: Smoothing the data
    processed_3 = smoothdata(processed_2,'gaussian');

    postprocessed = processed_3;
end

function features = featureExtract(table)
    %Average value in sensor responses
    sub_feature1 = mean(table);

    %RMS value in sensor responses
    sub_feature2= rms(table);

    %Standard Deviation in sensor responses
    sub_feature3 = std(table);

    %Peak-magnitude to RMS ratio in sensor responses
    %intialize array to temporary store feature
    sub_feature4 = zeros(1,11);
    for m = 1:1:11
        sub_feature4(m) = peak2rms(table(:,m));
    end

    %Skewness of each sensor response
    sub_feature5 =  skewness(table);

%     disp(table);
%     disp(sub_feature1);
%     disp(sub_feature2);
%     disp(sub_feature3);
%     disp(sub_feature4);
%     disp(sub_feature5);

    features = [sub_feature1,sub_feature2,sub_feature3,sub_feature4,...
        sub_feature5];

end

function readExport(decision, choice, training_model, testing_model, table)
    mean_header = ["Mean_MQ-2","Mean_MQ-3","Mean_MQ-4","Mean_MQ-5","Mean_MQ-
6","Mean_MQ-7",...
        "Mean_MQ-8","Mean_MQ-9","Mean_MQ-135","Mean_MQ-136","Mean_MQ-138"];

    RMS_header = ["RMS_MQ-2","RMS_MQ-3","RMS_MQ-4","RMS_MQ-5","RMS_MQ-6","RMS_MQ-
7","RMS_MQ-8",...
        "RMS_MQ-9","RMS_MQ-135","RMS_MQ-136","RMS_MQ-138"];

    Std_header = ["Std_MQ-2","Std_MQ-3","Std_MQ-4","Std_MQ-5","Std_MQ-6","Std_MQ-
7","Std_MQ-8",...
        "Std_MQ-9","Std_MQ-135","Std_MQ-136","Std_MQ-138"];

    Peak2rms_header = ["Peak2RMS-2","Peak2RMS-3","Peak2RMS-4","Peak2RMS-
5","Peak2RMS-6","Peak2RMS-7","Peak2RMS-8",...
```

```
        "Peak2RMS-9","Peak2RMS-135","Peak2RMS-136","Peak2RMS-138"];

    skew_header = ["Skew_MQ-2","Skew_MQ-3","Skew_MQ-4","Skew_MQ-5","Skew_MQ-
6","Skew_MQ-7","Skew_MQ-8",...
        "Skew_MQ-9","Skew_MQ-135","Skew_MQ-136","Skew_MQ-138"];

    classifier_header = "Classifier";

    header = 
[mean_header,RMS_header,Std_header,Peak2rms_header,skew_header,classifier_header];

    %adjust directory according to if user want to export Raw or
    %Preprocessed data
    if(choice == 1)
        training_model = strcat(training_model,'_Raw.xlsx');
        testing_model = strcat(testing_model,'_Raw.xlsx');

    elseif (choice == 2)
        training_model = strcat(training_model,'_Preprocessed.xlsx');
        testing_model = strcat(testing_model,'_Preprocessed.xlsx');
    end

    %determine if user wants to add to either training or testing dataset
    if (decision == 1)
        %read file
        %check if file is empty
        empty = isempty(readmatrix(training_model));

        %if empty, headers will be inputed
        if (empty==1)
            writematrix(header,training_model);
        end

        %export dataset and append to Training dataset Excel file
        writematrix(table,training_model,"WriteMode", "append");

    elseif(decision == 2)
        %read file
        %check if file is empty
        empty = isempty(readmatrix(testing_model));

        %if empty, headers will be inputed
        if (empty==1)
            writematrix(header,testing_model);
        end

        %export dataset and append to Testing dataset Excel file
        writematrix(table, testing_model, "WriteMode", "append");
    end

end

function graphing(Plotdirectory,table)
    for i = 1:1:11
        %create tile styled plot
        nexttile

        %plot colomn by colomn
        plot(table(:,i));
```

```
            %deciding title of each subplot
            if (i==1)
                title('MQ-2');
            elseif (i==2)
                title('MQ-3');
            elseif (i==3)
                title('MQ-4');
            elseif(i==4)
                title('MQ-5');
            elseif(i==5)
                title('MQ-6');
            elseif(i==6)
                title('MQ-7');
            elseif(i==7)
                title('MQ-8');
            elseif(i==8)
                title('MQ-9');
            elseif(i==9)
                title('MQ-135');
            elseif(i==10)
                title('MQ-136');
            elseif(i==11)
                title('MQ-138');
            end
        end

        %exporting plotted graph as png
        f = gcf; %to get the current figure and assign to 'f'
        %exportgraphics(f,Plotdirectory); %export the plot
    end
```