



**FACULTY OF ENGINEERING AND GREEN TECHNOLOGY**

**UGEA4333 Image Processing**

**Assignment Report: Project 1  
Ammonia water test using image processing**

**Prepared By:**

<b>Tan Li Kar</b>	<b>15AGB05449</b>
<b>Tan Ing Ming</b>	<b>17AGB05164</b>
<b>Yip Yuan Ting</b>	<b>13AGB05156</b>
<b>Lim Wan Xiang</b>	<b>15AGB01875</b>

**Date: 6 August 2019**

**Lecturer in charge: Humaira Nisar**

## 1.0 Introduction

Various substances and minerals exist in water, which is the main reason water has to be processed before it is consumable. Among the substances that exist, some can endanger the health of human or marine life. In this project, the main focus will be on the concentration of ammonia in the water. Ammonium ions are formed when ammonia dissolves in water, and both are toxic. Therefore, when the ammonia level reaches a certain threshold (more than 0.0 ppm), the water is said to be contaminated. Such issue is crucial in the field of water treatment. One of the common methods used to measure the ammonia level is the spectrophotometric method with the Nessler's reagent. In this project, the master test kit is used instead. This master test kit includes the ammonia liquid test kit (as shown in Figure 1) Aquarium Pharmaceuticals (n.d.).



Figure 1: The ammonia test kit, with bottle 1, bottle 2, and the test tube for the liquid specimen.

This kit is used because in this project, image processing will be implemented by comparing the color coding of water in the presence of ammonia to the color of the water after the test kit is applied to the water. However, ammonia itself is odorless and colorless, making it hard to distinguish the ammonia level without any further processes. Therefore, by adding 8 drops of sodium nitroprusside from bottle 1 (which can be obtained from the ammonia liquid test kit) Aquarium Pharmaceuticals (2014) and 8 drops of sodium hydroxide from bottle 2 (which can also be obtained from the ammonia liquid test kit)

Aquarium Pharmaceuticals (2018) and shaking it, the chemical reaction will cause the mixture to change its color. By comparing the color of the mixture with the color card provided in the kit, the ammonia level of the water specimen can be found out, as shown in Figure 2.



Figure 2: The comparison of the mixture and the color card. (Aqua-Fish.Net, n.d.)

Although human eye can be able to distinguish the comparison of the color, sometimes naked eye might make mistake, unlike image processing methods. Therefore, to obtain an accurate result, image processing will be carried out to obtain the measurement of the concentration of the ammonia in the water specimen according to the color card.

In this project, random forest algorithm is implemented. Random forest is made up of individual decision trees, in huge quantity. Decision tree is a type of decisive supporting tool. It is a flowchart-like structure that resembles the shape of a tree. Nodes in the decision trees contains a condition statement will split out into different outcomes and choices. These outcomes will then be treated as another different node and are further split again. These processes will loop until the outcome is not required to split anymore stated by Rajesh (2018).

These decision trees exist in the random forest and will carry out the decision making. The outcome of the majority of the decision trees will be the outcome of the

random forest model, which resembles of the voting process, with the decision trees as voters. The main idea of using the random forest model is to be able to produce a prediction that is agreed by majority of the decision trees, thus able to prevent individual errors from a few decision trees from affecting the final outcome of the random forest model Yiu (2019).

Navlani (2018) states that Naive Bayes is a statistical classification technique based on Bayes Theorem. It is one of the simplest supervised learning algorithms. Naive Bayes classifier is the fast, accurate and reliable algorithm. Naive Bayes classifiers have high accuracy and speed on large datasets.

Naive Bayes classifier assumes that the effect of a particular feature in a class is independent of other features. For example, a loan applicant is desirable or not depending on his/her income, previous loan and transaction history, age, and location. Even if these features are interdependent, these features are still considered independently. This assumption simplifies computation, and that's why it is considered as naive.

Random forest is suitable in training sets regarding classification, as in this project, by applying random forest, the accuracy in classifying the water samples collected according to the color of the mixture will be high.

## **2.0 Objective**

To determine the concentration of the ammonia in the water specimen with image processing methods of at least 2 algorithms.

### 3.0 Literature review

To start the project, some journals papers are referred to get a rough idea, which will help in planning the procedure of the project. A similar paper is taken as reference, titled “*Using a PC camera to determine the concentration of nitrite, ammonia nitrogen, sulfide, phosphate, and copper in water*” and written by B. Xin-Yue, L. Sheng, S. Wan-Gan. and G. Hong-Wen. (2018). In the paper, the concentration of several chemical substances are measured instead of just focusing on ammonia level. The paper proposes the method of getting the average RGB value in the center region of the image of the water specimen after getting the image with a PC camera. This is a good idea as the edge of the image and the background will not be included in the calculation algorithm for determining the concentration of chemical substances. The paper also mentions of a software designed by the authors to automatically compute the RGB parameters of the image after the image is taken.

After having a rough idea, the first thing to start with is the removal of the background, thus extracting only the test tube image, which is our interested part. This is where image segmentation takes place. Image segmentation is the technique which divides an image into multiple parts using digital image processing and analysing MathWorks (n.d.). In this topic, K. Bhargavi and S. Jyothi (2014) mentioned a few methods of global thresholding, which is histogram-based thresholding, iterative based threshold selection, Otsu’s method-based threshold selection, maximum correlation thresholding, clustering based threshold selection, and the multi-thresholding method. Given various methods, choices can be made according to the suitability in this project.

Upon getting the region of interest, noise have to be removed from the image. The noises in the image are due to the shades and designs, which will interfere with the results. Therefore, filters have to be implemented. B. Beddad and K. Hachemi (2018) suggested an improved median filter, which involves calculation of variance in selected areas within the filter. The area with the lowest value of variance amongst the selected areas will have its mean computed to replace the current processing pixel.

In extracting the color information, Y.A. Sari and S. Adinugroho (2017) did a similar research, which is the tomato ripeness clustering using 6-means algorithm based on v-channel otsu segmentation. Several processes in their paper is worth to be referred, such as the color transformation and the color feature clustering.

A similar topic in the paper published by H. M. Zawbaa1, M.am Hazman, M. Abbass, A. E. Hassanien (2014) proposes the random forest algorithm in automatic fruit classification. In the paper, the fruits recognition is mainly based on the shape and the color. Random Forest classifier is used in the classification process as it provides results with high accuracy in the classification of large dataset. The skills and methods for color recognition mentioned in this paper can be used as a good reference in the color recognition for the ammonia concentration.

To obtain more information regarding image classification using random forest, the paper titled “*An Improved Random Forest Classifier for Image Classification*” by B.X. Xu, Y.M. Ye and N. Lei (2012) is referred. The paper mentioned about a better version of a random forest model by running the normal random forest algorithm, but the final ensemble model will choose around 80% of the decision trees which is considered more accurate.

The paper published by S. Agarwal, N. Bhangale, K. Dhanure, S. Gavhane, V.A.Chakkarwar and Dr. M.B.Nagori, titled “Application of Colorimetry to determine Soil Fertility through Naive Bayes Classification Algorithm” works on a similar topic. Similar to this project, their project also requires skills in determining the concentration of a certain chemical substance in collected samples. Naive Bayes Classification Algorithm is applied in their project. The main idea of Naive Bayes is an algorithm that performs classification and in the process of classification, presumes that every occurrence of attributes are independent of each other.

#### 4.0 Methodology

In order to obtain the features of the images, the region of interest must be first extracted. The region of interest is cropped from the original images. The regions of interest are depicted with red rectangle as shown below.

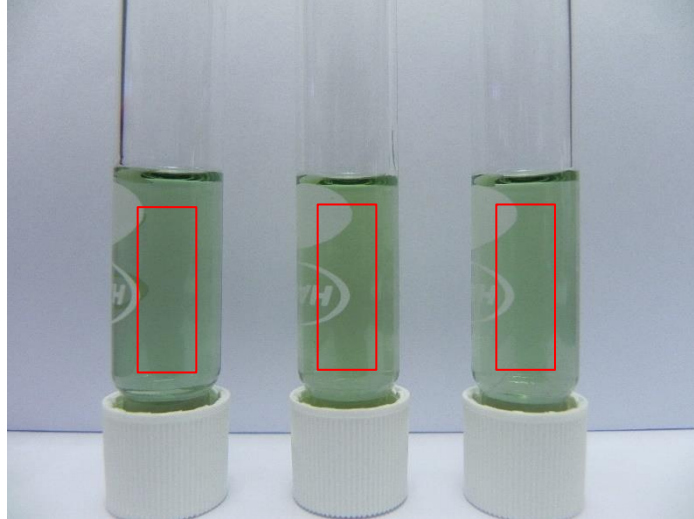


Figure 3: Region of interest depicted with red rectangle

After the images are cropped, the crop images are then filtered using a median filter to remove the designs on the test tubes. Next, the images are filtered using an average filter to remove the shading on the images. After the pre-processing is done, the mean and entropy of Red, Green, Blue, Hue, Saturation and Value are measured and saved into an excel file.

After obtaining the features of the images, the prediction of ammonia concentration is done using machine learning models which are Random Forest, Decision Tree and Naïve Bayes. The features measured are loaded and used to train the models. Each day have a total of 19 samples totalling at 38 samples for both days. Due to the small sample size, K-Fold validation is used to measure the accuracy of the models. In this assignment 19-Fold validation was used. When the models are trained for the days separately, this means that

for each iteration 18 samples are randomly selected for training the machine learning models while 1 sample is randomly selected for testing the machine learning models. When the models are trained for both days simultaneously, 36 samples are randomly selected for training and 2 are randomly selected for testing. The accuracy for the machine learning models is recorded.

MATLAB was used to extract the region of interest in those images, process it and put them into an excel file. Next, Spyder was used to implement three types of machine learning classifiers to those image data in the excel file, Spyder is an open source cross-platform integrated development environment for scientific programming in the Python language. The classifiers that were used are random forest, Decision Tree and Naïve Bayes. The results and accuracy got from these classifiers was then placed in the results section below.



5.0 Results

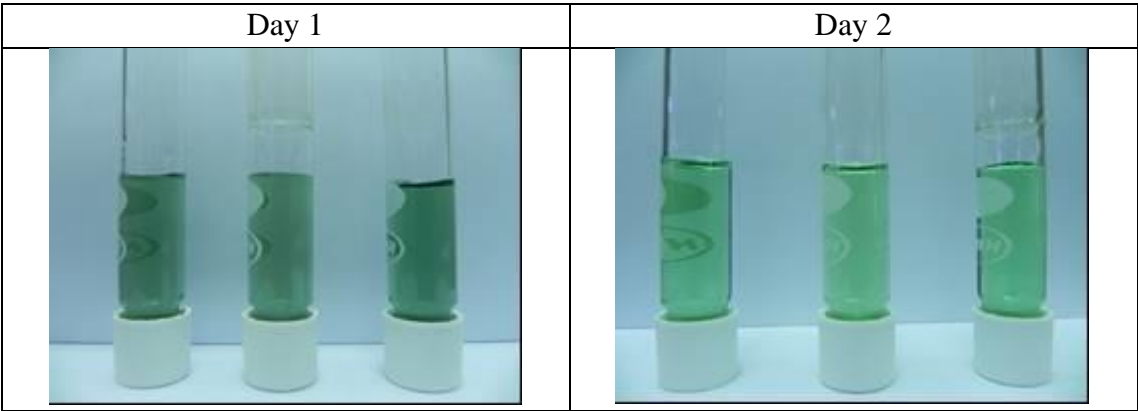


Figure 4: The original images of water samples in day 1 and day 2 with ammonia level of 0.25 ppm.

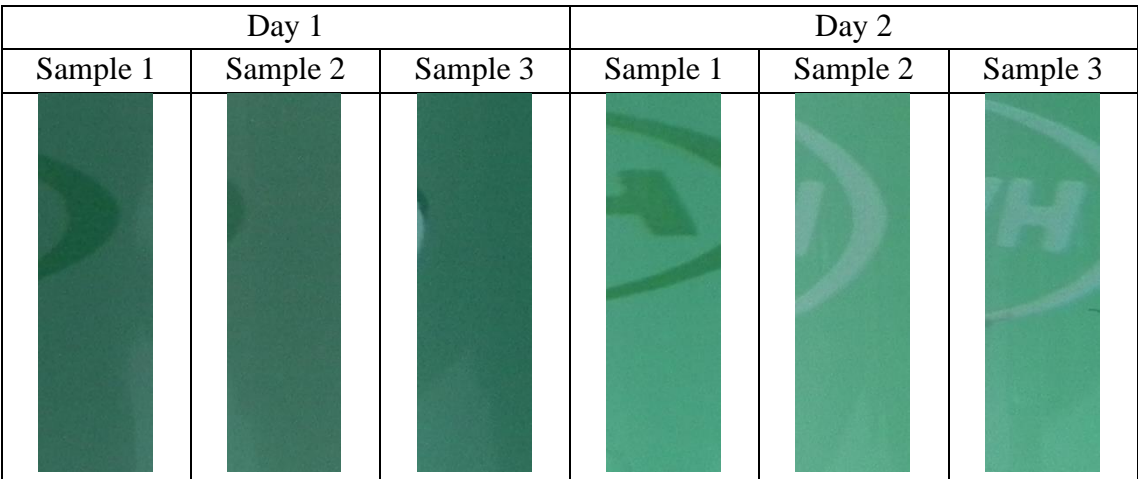


Figure 5: Cropped imaged for water samples with ammonia level of 0.25 ppm in day 1 and day 2.







Day 1			Day 2		
Sample 1	Sample 2	Sample 3	Sample 1	Sample 2	Sample 3
					

Figure 6: Median filtered images for water samples with ammonia level of 0.25 ppm in day 1 and day 2.







Day 1			Day 2		
Sample 1	Sample 2	Sample 3	Sample 1	Sample 2	Sample 3
					

Figure 7: Average filtered images for water samples with ammonia level of 0.25 ppm in day 1 and day 2.

Table 1: Mean and Entropy of Red, Green and Blue (RGB) channel and Hue, Saturation and Value (HSV) channel for water sample with ammonia level of 0.25 ppm in day 1.

0.25ppm, Day 1		Sample		
		1	2	3
Mean	Red	49.3047	60.0638	47.1130
	Green	102.0179	110.5833	109.5690
	Blue	84.2292	92.1270	89.3742
	Hue	0.4438	0.4391	0.4461
	Saturation	0.5171	0.4569	0.5708
	Value	0.4001	0.4337	0.4297
Entropy	Red	3.4606	2.5311	3.6884
	Green	3.3239	2.9142	3.6351
	Blue	3.6351	2.7528	3.8600
	Hue	1.3834	1.3407	1.3323
	Saturation	3.5763	2.1405	4.1161
	Value	3.3239	2.9142	3.6351

Table 2: Mean and Entropy of RGB channel and HSV channel for water sample with ammonia level of 0.25 ppm in day 2.

0.25 ppm, Day 2		Sample		
		1	2	3
Mean	Red	69.8918	86.8897	80.9609
	Green	154.1516	163.4640	161.4777
	Blue	120.8679	131.8310	130.6411
	Hue	0.4341	0.4312	0.4362
	Saturation	0.5469	0.4685	0.4986
	Value	0.6045	0.6410	0.6332

Entropy	Red	3.4607	3.6099	3.5811
	Green	3.9255	3.8346	4.0755
	Blue	3.9654	3.8765	4.0030
	Hue	1.7577	1.3442	1.1647
	Saturation	2.9062	3.4860	3.2520
	Value	3.9255	3.8346	4.0755

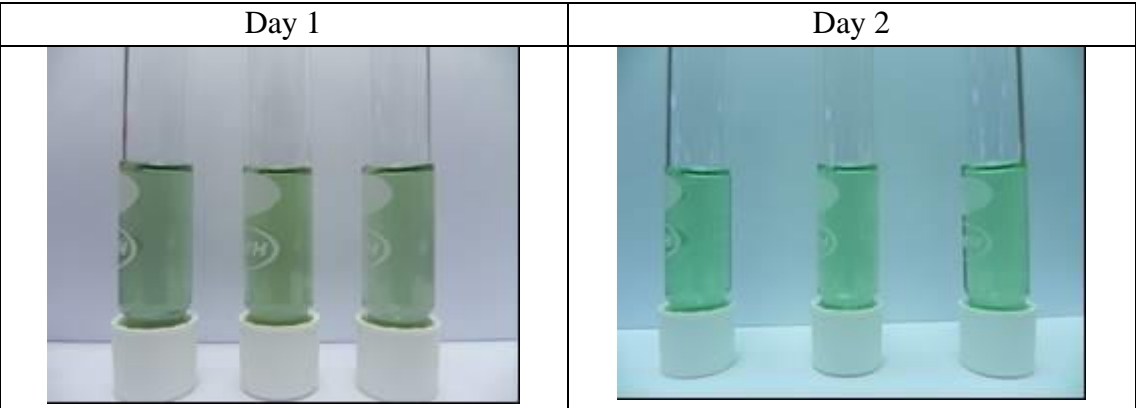


Figure 8: The original images of water samples in day 1 and day 2 with ammonia level of 0.5 ppm.

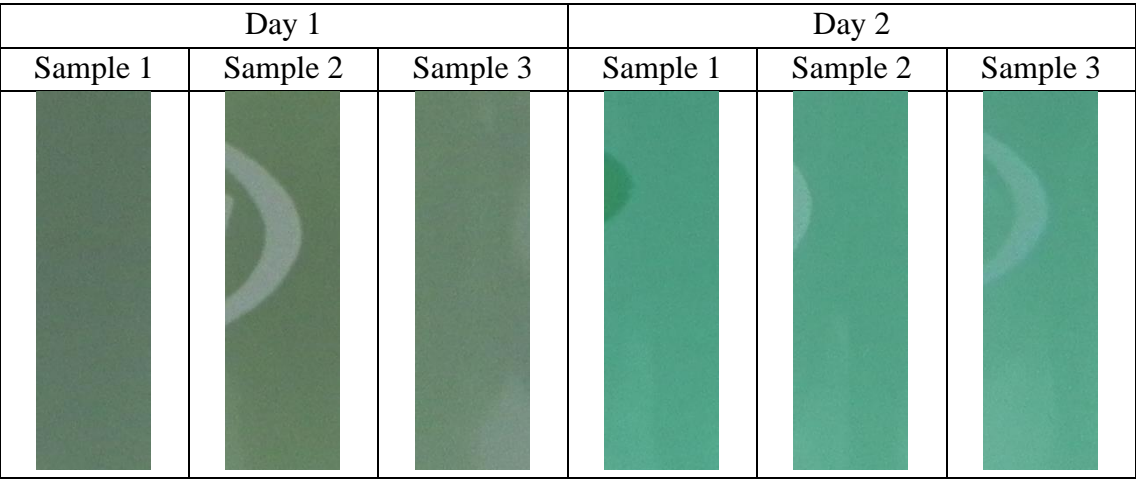


Figure 9: Cropped imaged for water samples with ammonia level of 0.5 ppm in day 1 and day 2.







Day 1			Day 2		
Sample 1	Sample 2	Sample 3	Sample 1	Sample 2	Sample 3
					

Figure 10: Median filtered images for water samples with ammonia level of 0.5 ppm in day 1 and day 2.







Day 1			Day 2		
Sample 1	Sample 2	Sample 3	Sample 1	Sample 2	Sample 3
					

Figure 11: Average filtered images for water samples with ammonia level of 0.5 ppm in day 1 and day 2.

Table 3: Mean and Entropy of RGB channel and HSV channel for water sample with ammonia level of 0.5 ppm in day 1.

0.5ppm, Day 1		Sample		
		1	2	3
Mean	Red	94.1440	98.7241	105.8425
	Green	118.8877	124.6162	134.0304
	Blue	99.6941	96.4226	111.4700
	Hue	0.3705	0.3200	0.3666
	Saturation	0.2080	0.2263	0.2104
	Value	0.4662	0.4887	0.5256
Entropy	Red	2.4390	2.4576	3.0532
	Green	2.9243	3.0077	3.1414
	Blue	2.8009	3.0232	3.6833
	Hue	2.0765	2.2617	2.6681
	Saturation	2.3333	3.5735	2.4179
	Value	2.9243	3.0077	3.1414

Table 4: Mean and Entropy of RGB channel and HSV channel for water sample with ammonia level of 0.5 ppm in day 2.

0.5 ppm, Day 2		Sample		
		1	2	3
Mean	Red	78.1933	84.2378	88.3783
	Green	159.3945	162.3048	164.3890
	Blue	132.4300	135.2690	138.5847
	Hue	0.4447	0.4423	0.4434
	Saturation	0.5096	0.4811	0.4627
	Value	0.6251	0.6365	0.6447
Entropy	Red	3.1174	3.2101	4.1204
	Green	3.6927	3.5057	4.1350

	Blue	3.6025	3.6136	4.2398
	Hue	0.9970	0.5319	0.8709
	Saturation	2.3212	2.5969	3.5552
	Value	3.6927	3.5057	4.1350

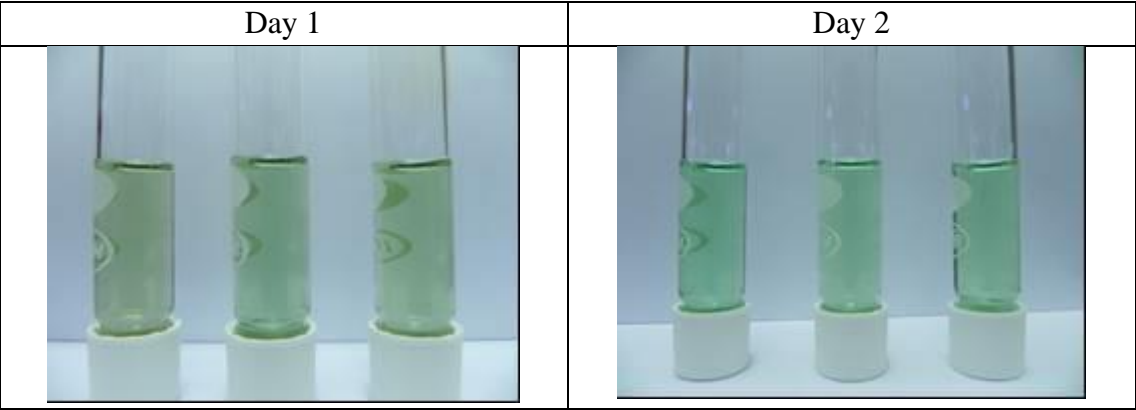


Figure 12: The original images of water samples in day 1 and day 2 with ammonia level of 1.0 ppm.

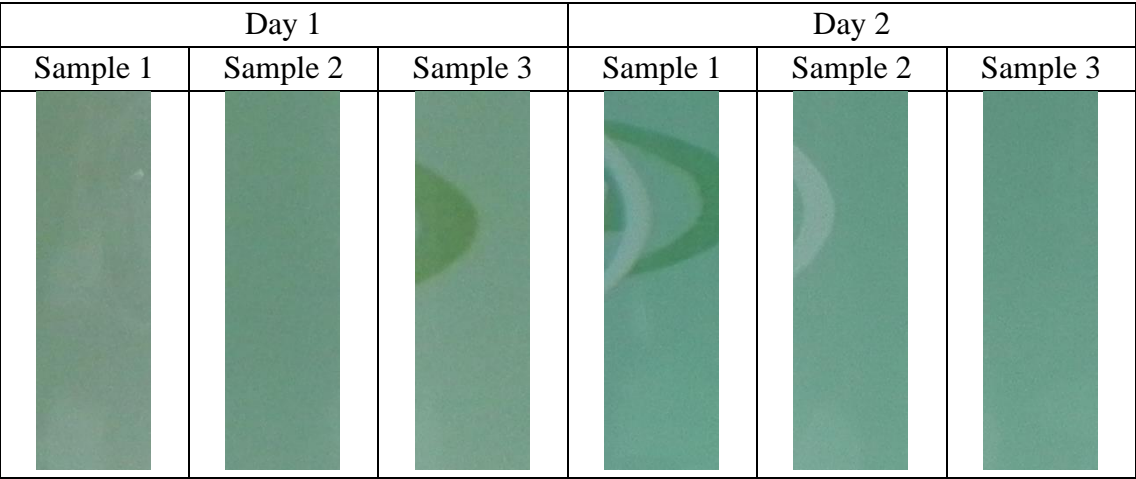


Figure 13: Cropped imaged for water samples with ammonia level of 1.0 ppm in day 1 and day 2.







Day 1			Day 2		
Sample 1	Sample 2	Sample 3	Sample 1	Sample 2	Sample 3
					

Figure 14: Median filtered images for water samples with ammonia level of 1.0 ppm in day 1 and day 2.







Day 1			Day 2		
Sample 1	Sample 2	Sample 3	Sample 1	Sample 2	Sample 3
					

Figure 15: Average filtered images for water samples with ammonia level of 1.0 ppm in day 1 and day 2.



Table 5: Mean and Entropy of RGB channel and HSV channel for water sample with ammonia level of 1.0 ppm in day 1.

1.0 ppm, Day 1		Sample		
		1	2	3
Mean	Red	119.7828	106.1744	115.0007
	Green	149.4544	148.3683	154.5602
	Blue	130.6419	126.4114	132.4243
	Hue	0.3943	0.4133	0.4068
	Saturation	0.1986	0.2844	0.2562
	Value	0.5861	0.5818	0.6061
Entropy	Red	3.3810	2.4587	3.5693
	Green	3.4983	2.1906	3.3565
	Blue	3.9856	3.2896	4.0089
	Hue	2.7920	2.3995	2.4730
	Saturation	2.3166	2.4010	3.0407
	Value	3.4983	2.1906	3.3565

Table 6: Mean and Entropy of RGB channel and HSV channel for water sample with ammonia level of 1.0 ppm in day 2.

1.0 ppm, Day 2		Sample		
		1	2	3
Mean	Red	98.2470	106.5789	104.1489
	Green	152.9339	157.1882	155.2947
	Blue	134.0713	138.2103	136.0555
	Hue	0.4425	0.4375	0.4373
	Saturation	0.3578	0.3222	0.3297
	Value	0.5997	0.6164	0.6090

Entropy	Red	3.7597	3.9520	3.8982
	Green	3.7545	3.7448	3.7888
	Blue	4.0291	3.8480	3.8775
	Hue	1.8066	0.9814	0.8196
	Saturation	3.1290	3.3871	3.1990
	Value	3.7545	3.7448	3.7888

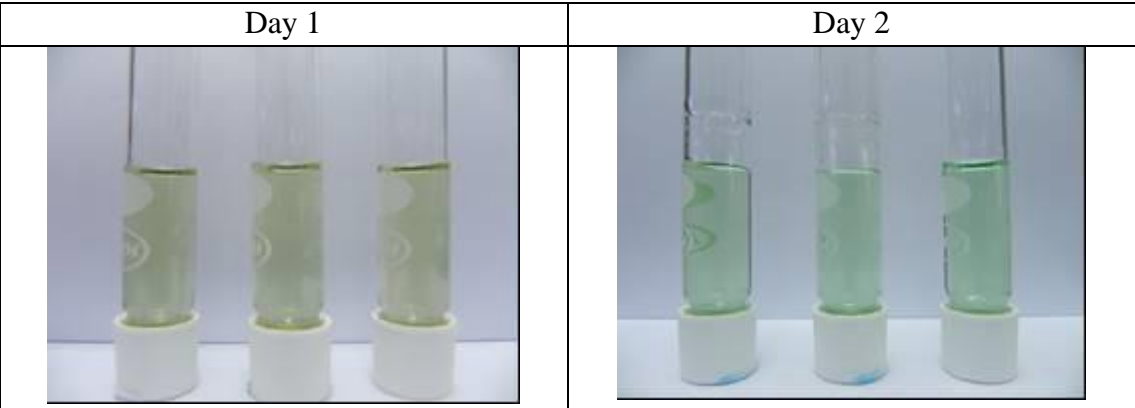


Figure 16: The original images of water samples in day 1 and day 2 with ammonia level of 2.0 ppm.

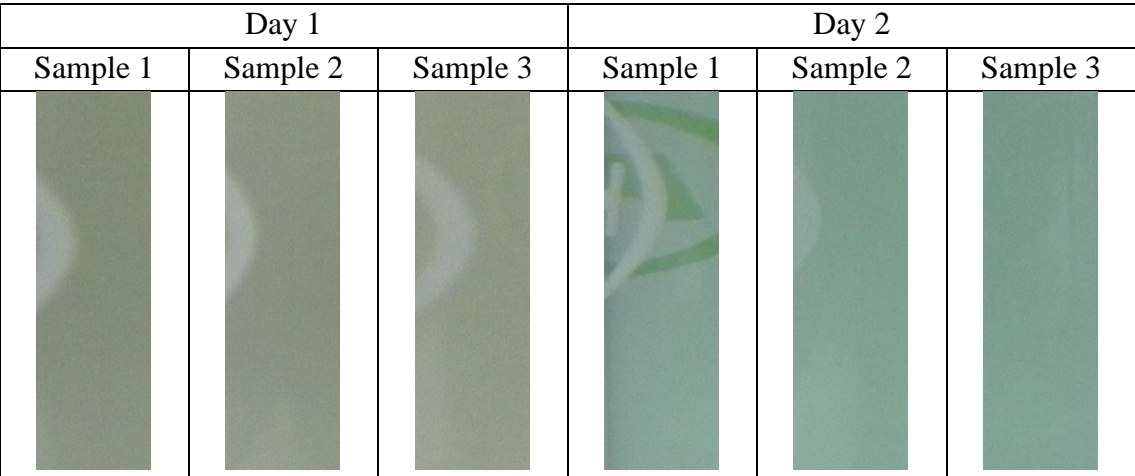


Figure 17: Cropped imaged for water samples with ammonia level of 2.0 ppm in day 1 and day 2.







Day 1			Day 2		
Sample 1	Sample 2	Sample 3	Sample 1	Sample 2	Sample 3
					

Figure 18: Median filtered images for water samples with ammonia level of 2.0 ppm in day 1 and day 2.







Day 1			Day 2		
Sample 1	Sample 2	Sample 3	Sample 1	Sample 2	Sample 3
					

Figure 19: Average filtered images for water samples with ammonia level of 2.0 ppm in day 1 and day 2.

Table 7: Mean and Entropy of RGB channel and HSV channel for water sample with ammonia level of 2.0 ppm in day 1.

2.0 ppm, Day 1		Sample		
		1	2	3
Mean	Red	134.0710	140.0317	146.2789
	Green	142.9470	148.1207	154.4380
	Blue	124.1682	131.1205	135.9861
	Hue	0.2457	0.2463	0.2406
	Saturation	0.1315	0.1149	0.1196
	Value	0.5606	0.5809	0.6056
Entropy	Red	2.7012	2.9200	3.1416
	Green	2.5293	2.9573	3.1257
	Blue	3.4568	3.4020	3.6262
	Hue	2.2861	2.5794	2.4359
	Saturation	2.6497	3.0409	3.0523
	Value	2.5293	2.9573	3.1257

Table 8: Mean and Entropy of RGB channel and HSV channel for water sample with ammonia level of 2.0 ppm in day 2.

2.0 ppm, Day 2		Sample		
		1	2	3
Mean	Red	130.8526	129.0192	125.5431
	Green	157.3995	161.3431	158.8897
	Blue	146.2860	146.5775	144.7709
	Hue	0.4300	0.4238	0.4294
	Saturation	0.1686	0.2004	0.2099
	Value	0.6173	0.6327	0.6231

Entropy	Red	3.4713	3.8688	3.5074
	Green	3.8934	3.8971	3.6508
	Blue	3.9773	4.0294	3.7592
	Hue	2.9097	1.9021	1.7819
	Saturation	2.2170	2.0884	2.1462
	Value	3.8934	3.8971	3.6508

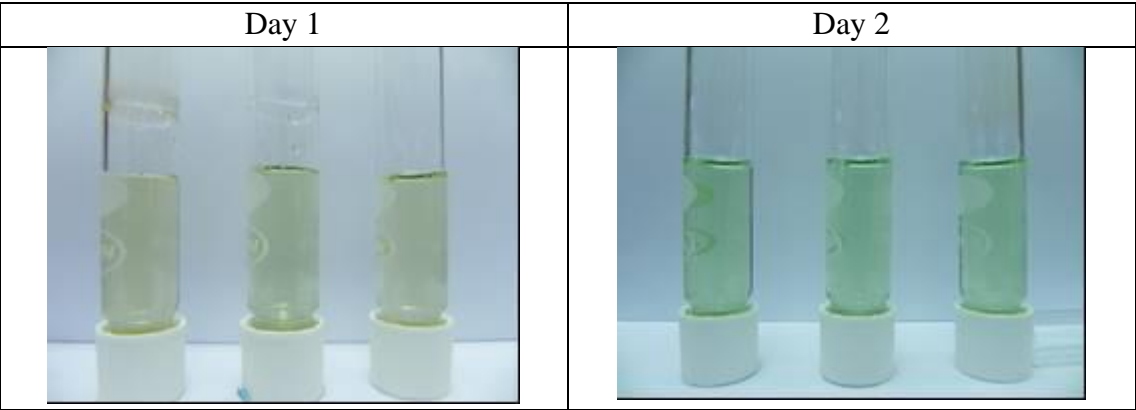


Figure 20: The original images of water samples in day 1 and day 2 with ammonia level of 4.0 ppm.

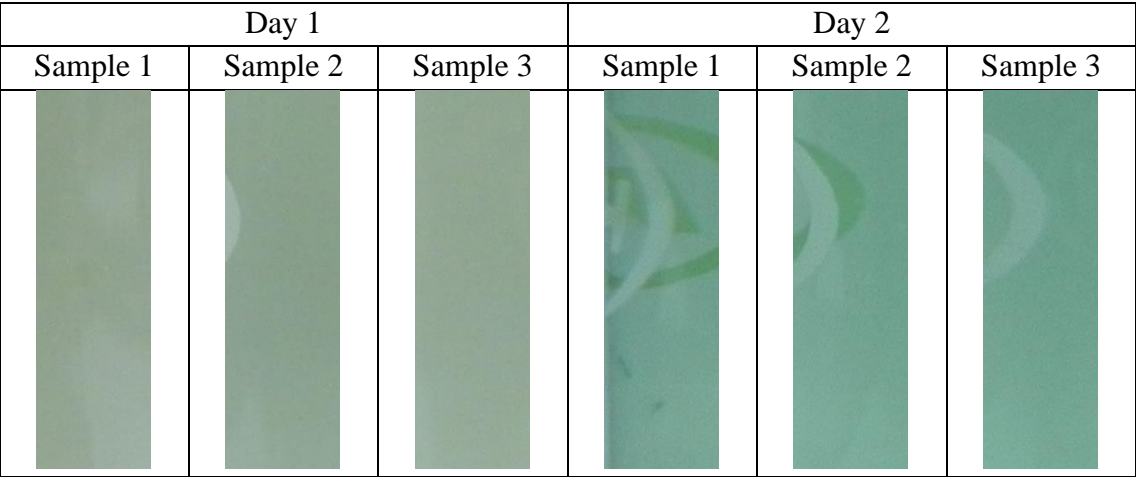


Figure 21: Cropped imaged for water samples with ammonia level of 4.0 ppm in day 1 and day 2.







Day 1			Day 2		
Sample 1	Sample 2	Sample 3	Sample 1	Sample 2	Sample 3
					

Figure 22: Median filtered images for water samples with ammonia level of 4.0 ppm in day 1 and day 2.







Day 1			Day 2		
Sample 1	Sample 2	Sample 3	Sample 1	Sample 2	Sample 3
					

Figure 23: Average filtered images for water samples with ammonia level of 4.0 ppm in day 1 and day 2.

Table 9: Mean and Entropy of RGB channel and HSV channel for water sample with ammonia level of 4.0 ppm in day 1.

4.0 ppm, Day 1		Sample		
		1	2	3
Mean	Red	146.9284	138.9586	146.5520
	Green	164.2675	162.1814	167.7538
	Blue	146.4088	143.5041	149.3107
	Hue	0.3294	0.3659	0.3550
	Saturation	0.1121	0.1432	0.1264
	Value	0.6442	0.6360	0.6579
Entropy	Red	3.4267	2.8964	2.5817
	Green	3.4752	2.7857	2.3991
	Blue	3.9989	3.2119	3.1050
	Hue	2.8262	2.4348	1.9575
	Saturation	2.7350	1.7709	1.5361
	Value	3.4752	2.7857	2.3991

Table 10: Mean and Entropy of RGB channel and HSV channel for water sample with ammonia level of 4.0 ppm in day 2.

4.0 ppm, Day 2		Sample		
		1	2	3
Mean	Red	111.6041	114.8487	111.8375
	Green	157.7460	163.7182	161.8491
	Blue	141.5325	145.5041	143.0498
	Hue	0.4414	0.4379	0.4374
	Saturation	0.2927	0.2988	0.3093
	Value	0.6186	0.6420	0.6347

Entropy	Red	3.8689	4.0319	3.9701
	Green	3.8990	3.8741	3.8802
	Blue	3.9293	4.0783	3.9643
	Hue	2.3404	1.2644	0.9848
	Saturation	3.1414	3.3050	3.2568
	Value	3.8990	3.8741	3.8802

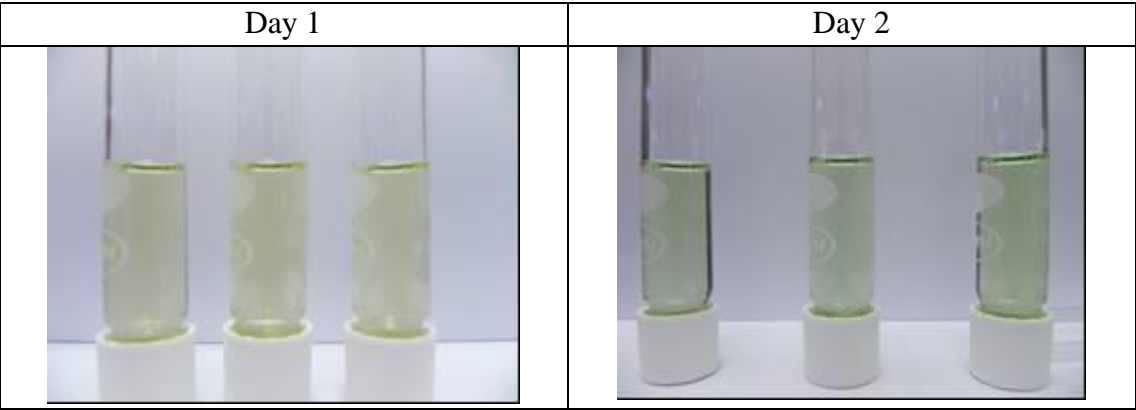


Figure 24: The original images of water samples in day 1 and day 2 with ammonia level of 8.0 ppm.

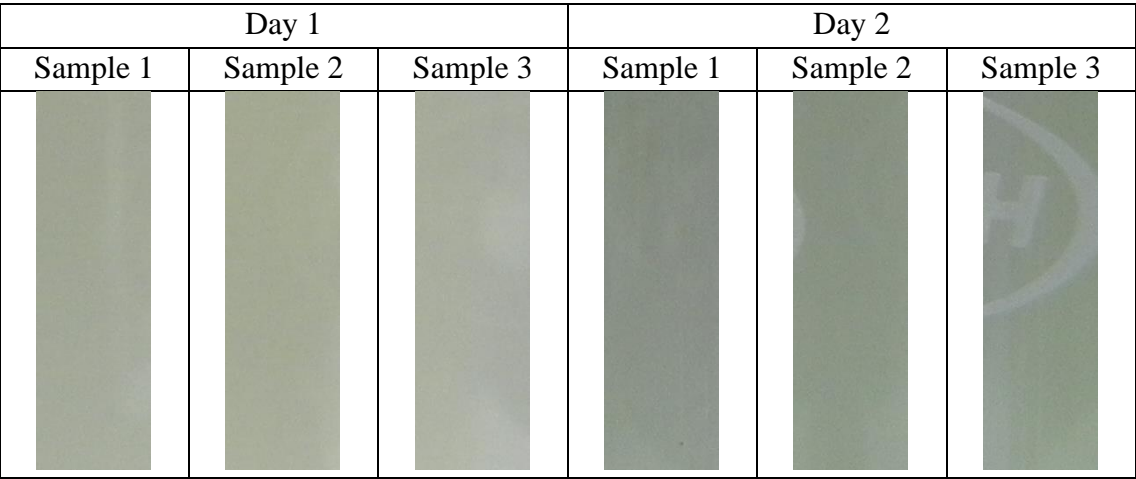


Figure 25: Cropped imaged for water samples with ammonia level of 8.0 ppm in day 1 and day 2.









Day 1			Day 2		
Sample 1	Sample 2	Sample 3	Sample 1	Sample 2	Sample 3
					

Figure 26: Median filtered images for water samples with ammonia level of 8.0 ppm in day 1 and day 2.







Day 1			Day 2		
Sample 1	Sample 2	Sample 3	Sample 1	Sample 2	Sample 3
					

Figure 27: Average filtered images for water samples with ammonia level of 8.0 ppm in day 1 and day 2.

Table 11: Mean and Entropy of RGB channel and HSV channel for water sample with ammonia level of 8.0 ppm in day 1.

8.0 ppm, Day 1		Sample		
		1	2	3
Mean	Red	163.4984	166.2228	171.0186
	Green	168.2209	170.8742	174.8680
	Blue	152.8862	151.0957	162.4971
	Hue	0.2182	0.2060	0.2191
	Saturation	0.0912	0.1158	0.0708
	Value	0.6597	0.6701	0.6858
Entropy	Red	3.1263	2.5313	3.2077
	Green	3.2494	2.3006	3.0897
	Blue	3.5513	3.4010	3.7825
	Hue	2.5378	1.9938	2.7903
	Saturation	2.5256	2.7462	2.9323
	Value	3.2494	2.3006	3.0897

Table 12: Mean and Entropy of RGB channel and HSV channel for water sample with ammonia level of 8.0 ppm in day 2.

8.0 ppm, Day 2		Sample		
		1	2	3
Mean	Red	137.0555	138.9103	139.6010
	Green	145.0674	149.5907	150.5373
	Blue	134.3709	134.3146	137.8894
	Hue	0.2919	0.2831	0.3138
	Saturation	0.0738	0.1021	0.0843
	Value	0.5689	0.5866	0.5903

Entropy	Red	3.6363	3.4168	3.5892
	Green	3.8439	3.7086	3.9438
	Blue	3.9505	3.5900	3.7403
	Hue	2.8526	2.2572	3.2127
	Saturation	2.6695	1.7693	3.6264
	Value	3.8439	3.7086	3.9438

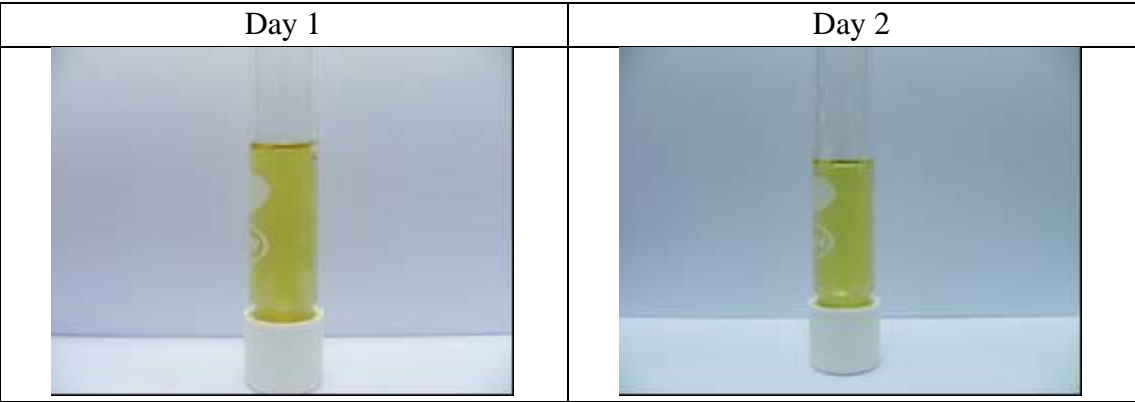


Figure 28: The original images of control sample (0 ppm) in day 1 and day 2

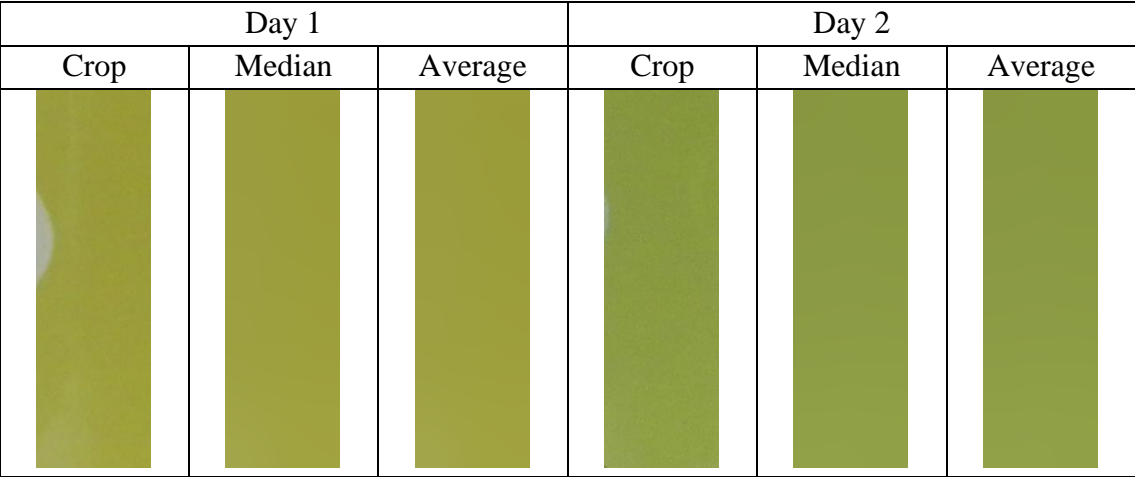


Figure 29: Cropped, median filtered and average filtered images for control samples in day 1 and day 2.

Table 13: Mean and Entropy of RGB channel and HSV channel for control samples in day 1 and day 2.

		Day	
		1	2
Mean	Red	156.8501	139.3733
	Green	158.3736	155.2813
	Blue	61.5044	66.9153
	Hue	0.1693	0.1967
	Saturation	0.6119	0.5692
	Value	0.6211	0.6089
Entropy	Red	2.8678	3.1464
	Green	3.1990	3.2350
	Blue	3.6249	3.2568
	Hue	0.3580	0.2437
	Saturation	3.8237	3.0644
	Value	3.1990	3.2350

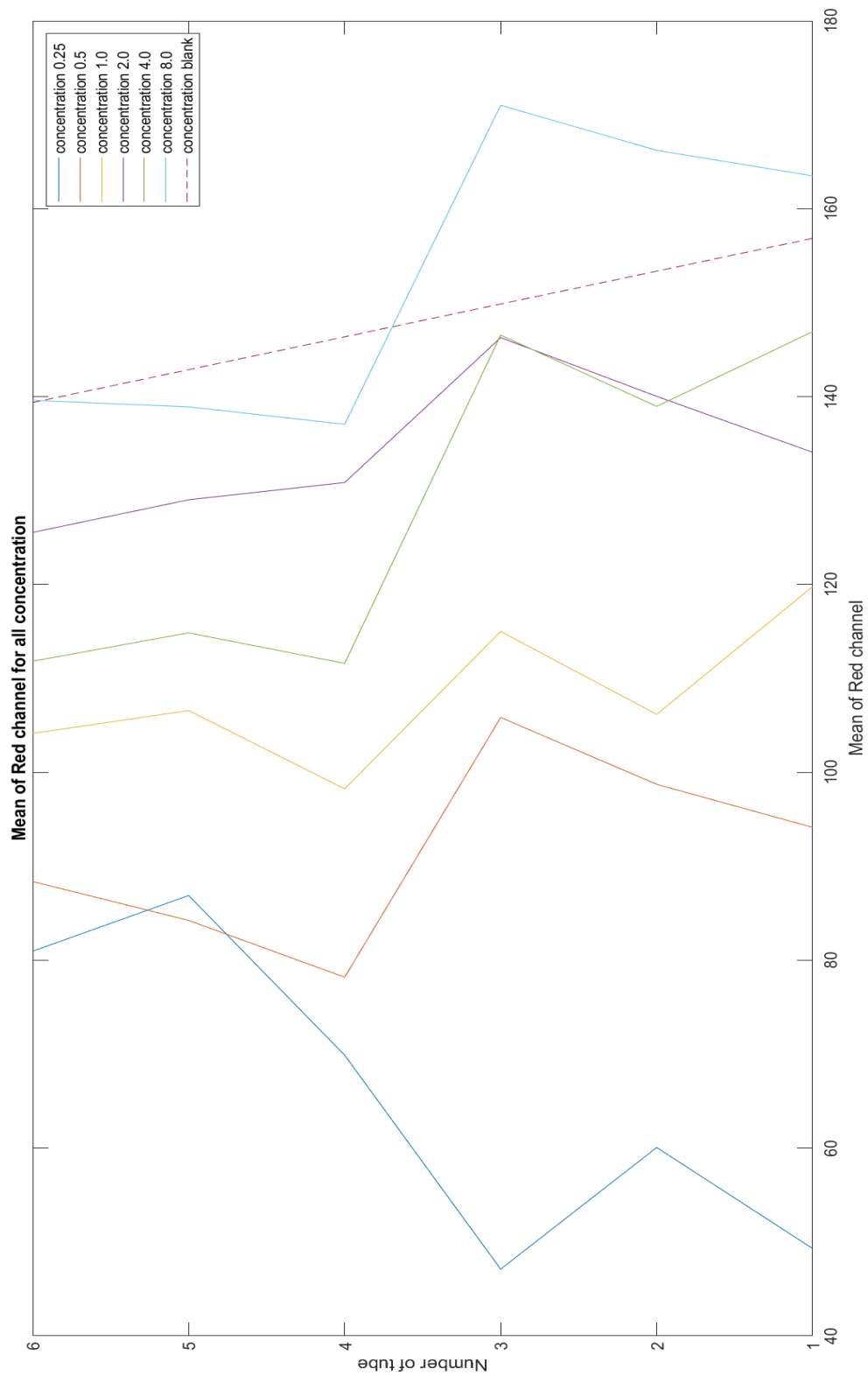


Figure 30: Graph showing the Mean of Red channel for images with different concentration of ammonia. (Tube 1 to 3 are samples for day 1 and Tube 4 to 6 are samples for day 2.)

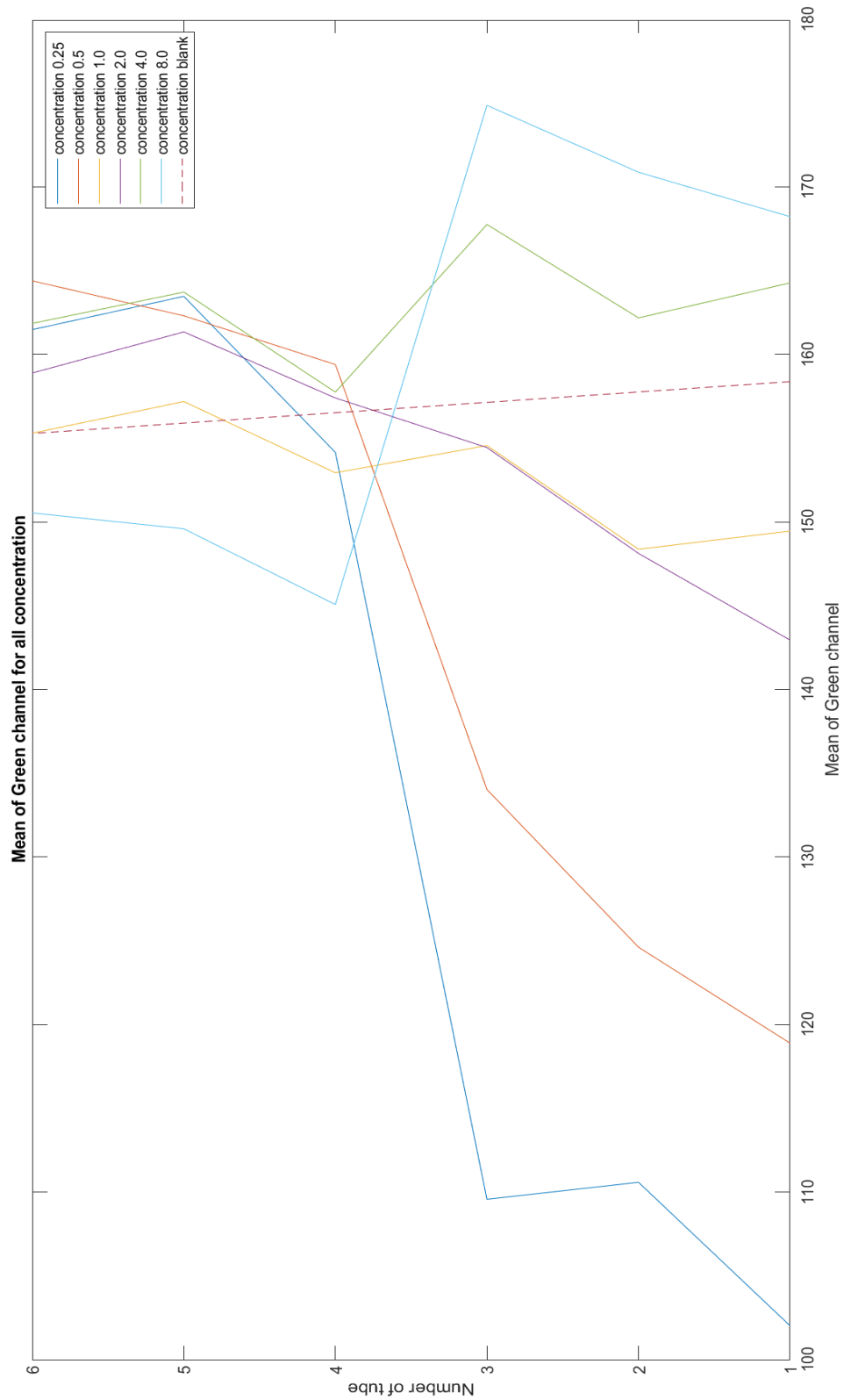


Figure 31: Graph showing the Mean of Green channel for images with different concentration of ammonia.

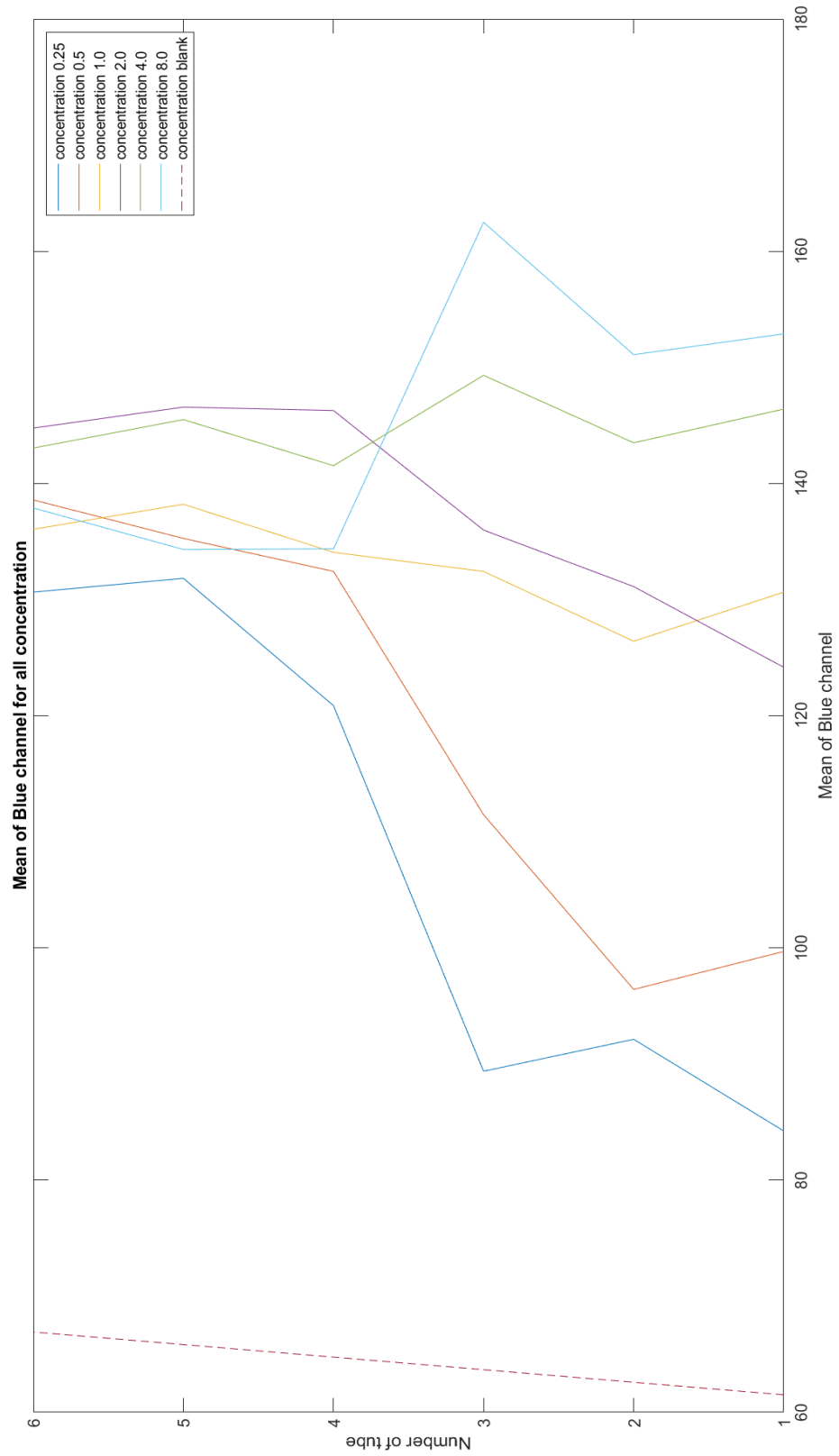


Figure 32: Graph showing the Mean of Blue channel for images with different concentration of ammonia.

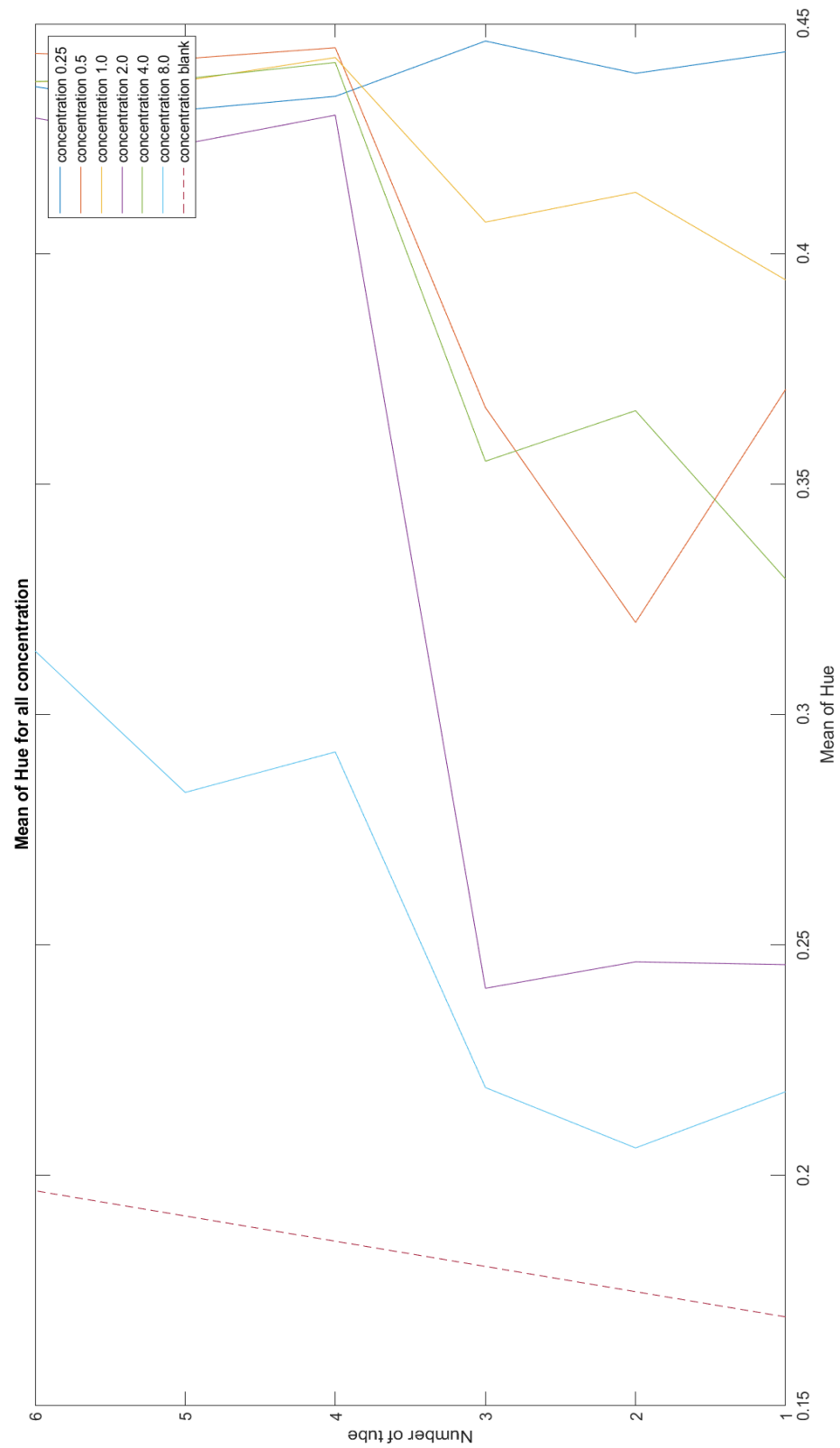


Figure 33: Graph showing the Mean of Hue for images with different concentration of ammonia.



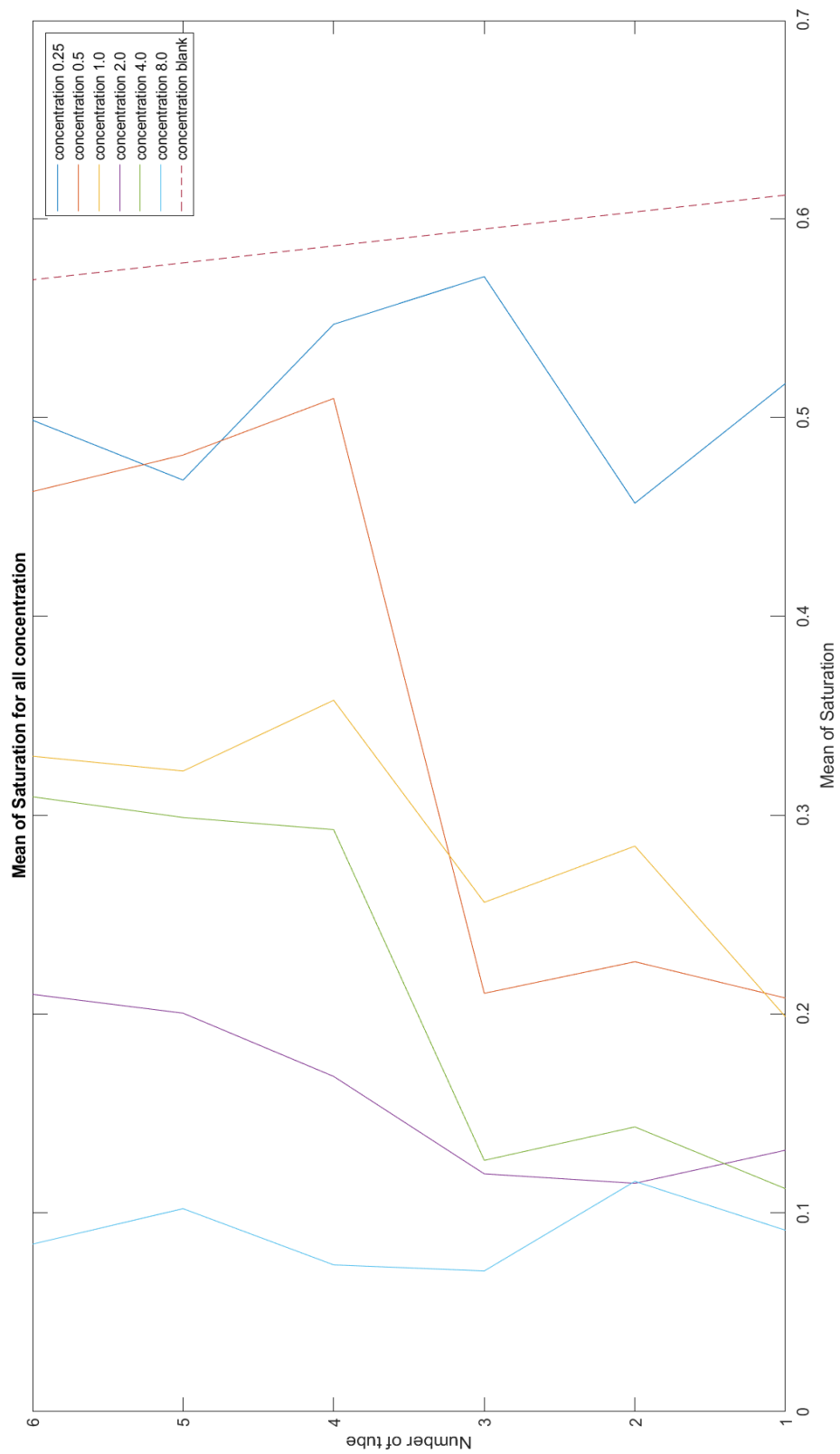


Figure 34: Graph showing the Mean of Saturation for images with different concentration of ammonia.

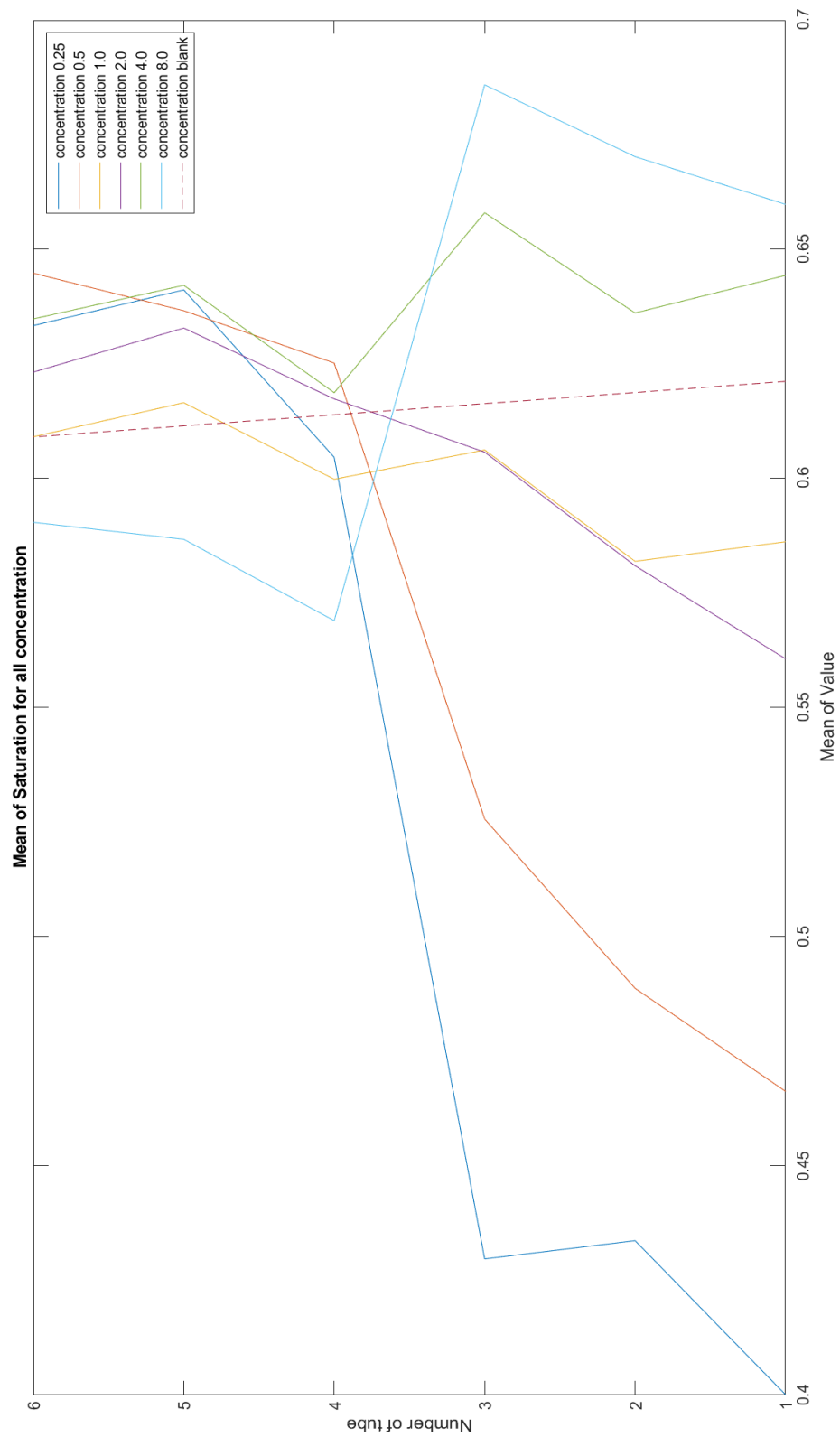


Figure 35: Graph showing the Mean of Value for images with different concentration of ammonia.

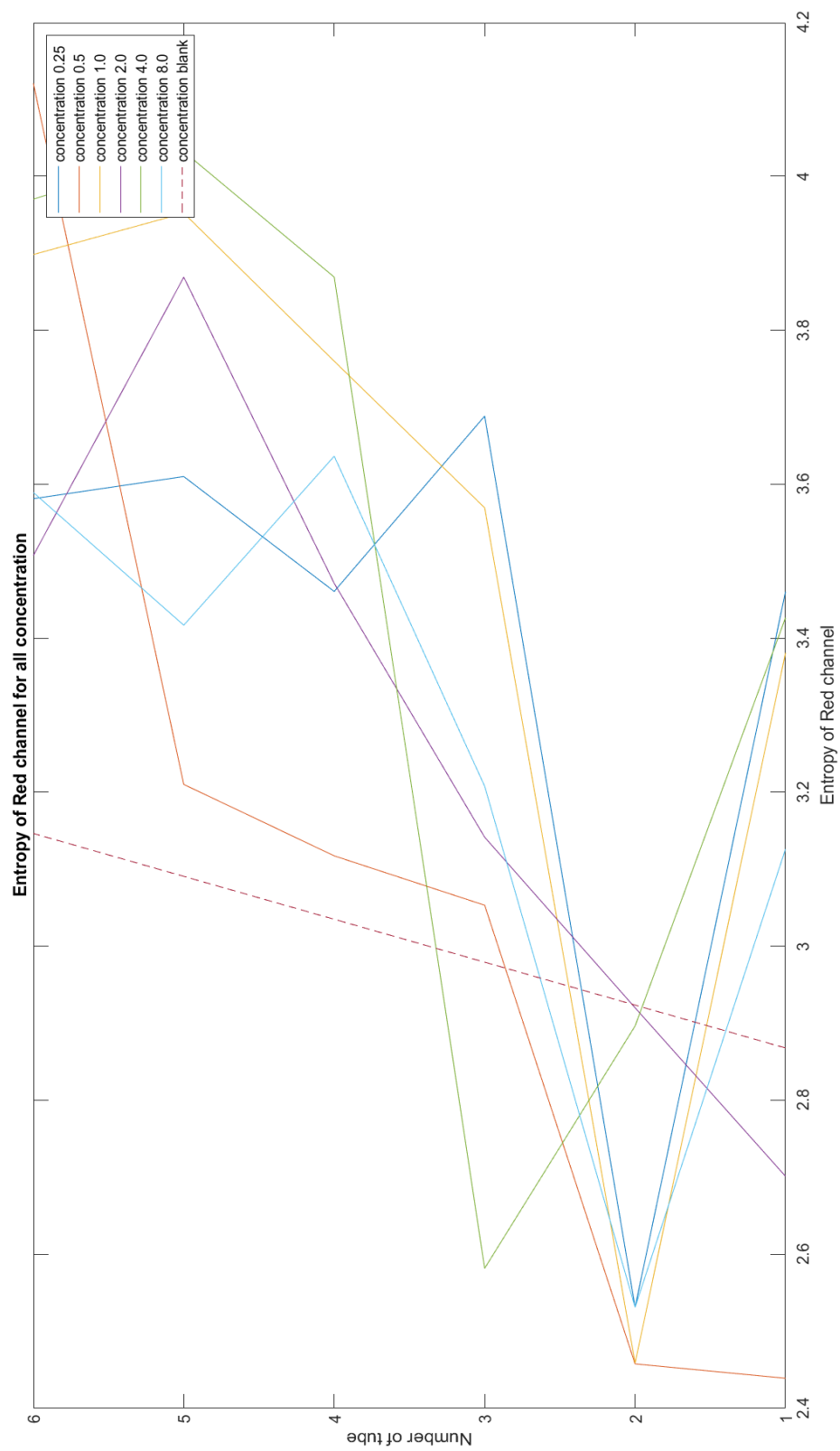


Figure 36: Graph showing the Entropy of Red channel for images with different concentration of ammonia.

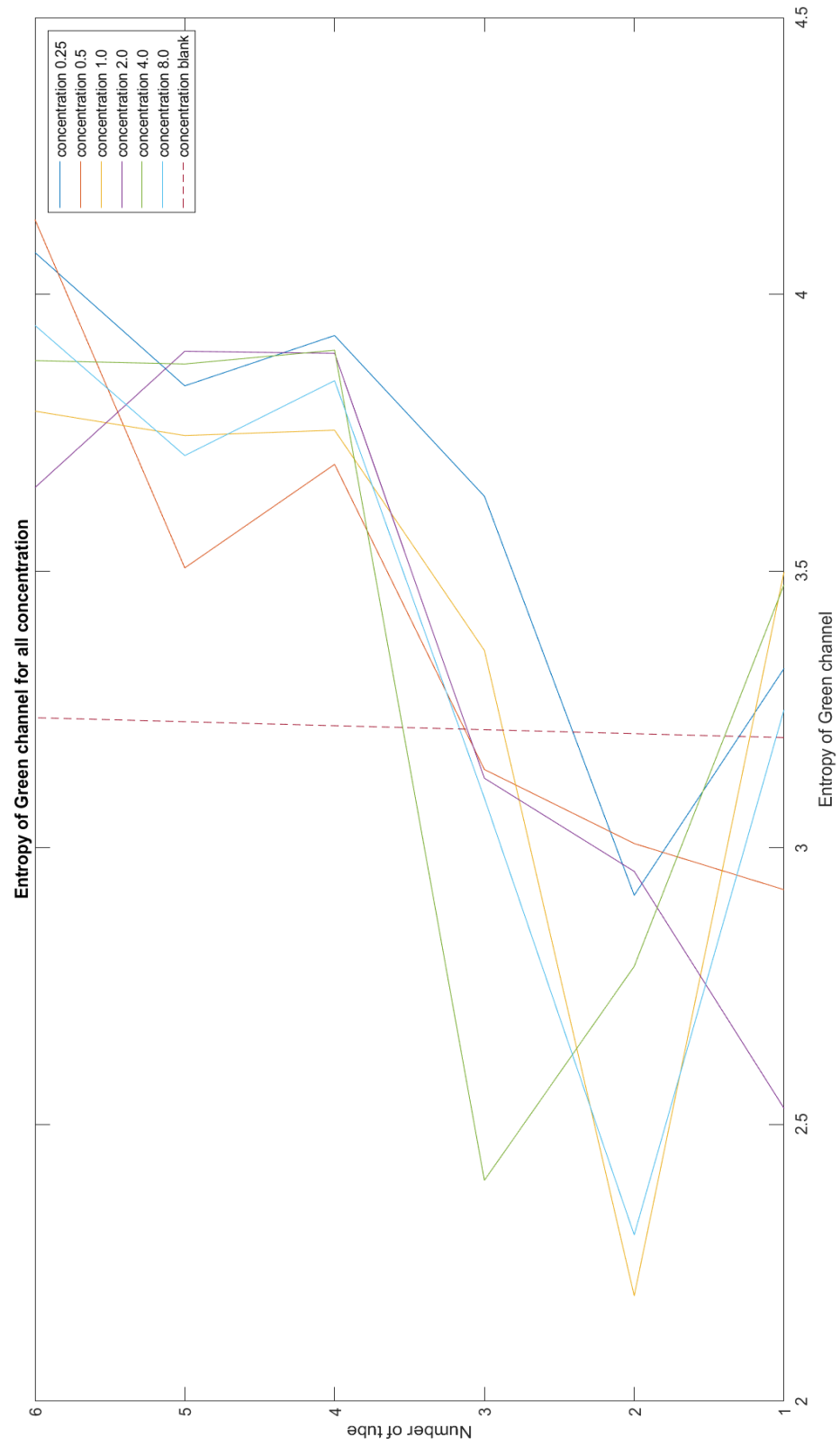


Figure 37: Graph showing the Entropy of Green channel for images with different concentration of ammonia.

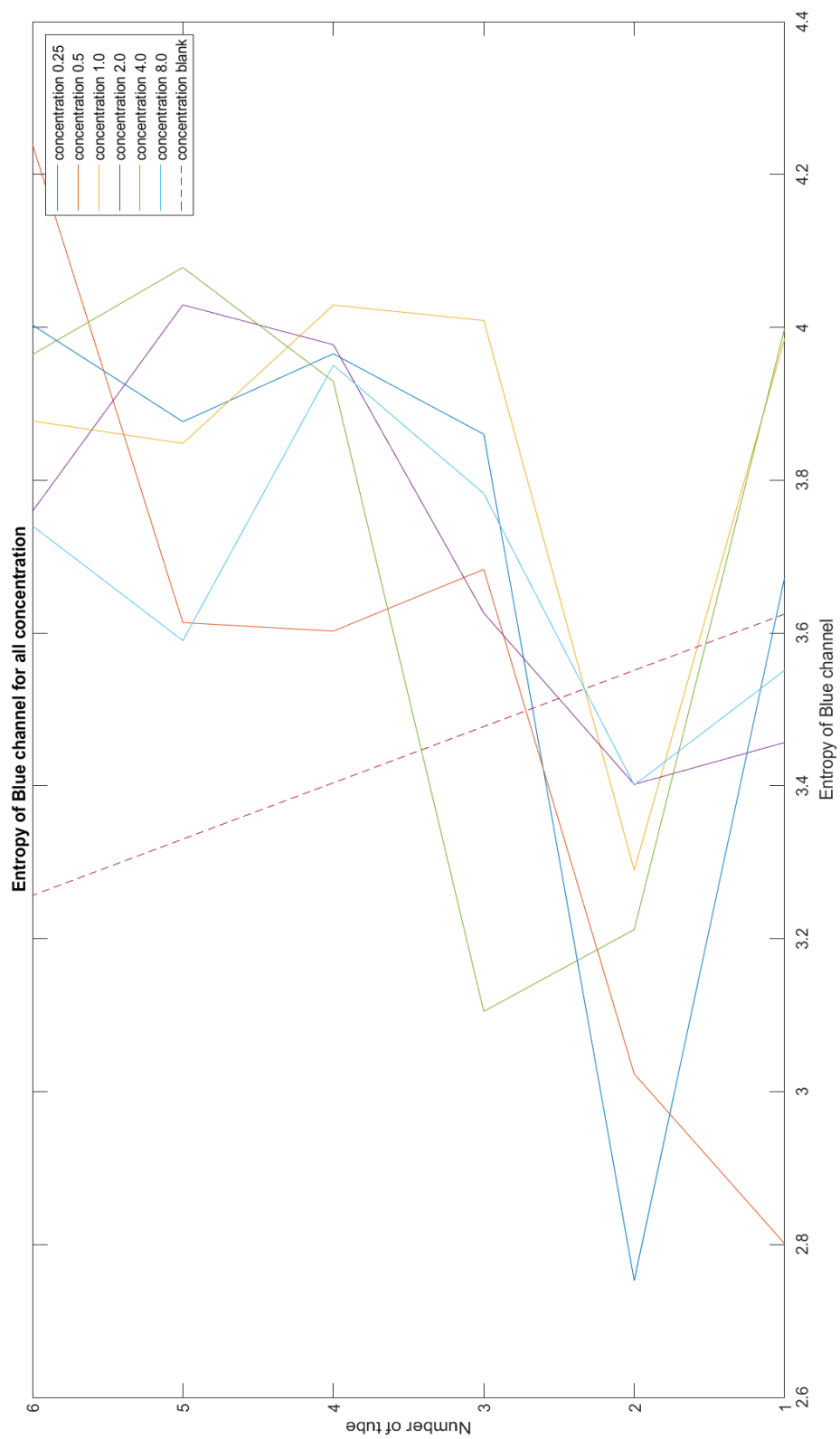


Figure 38: Graph showing the Entropy of Blue channel for images with different concentration of ammonia.

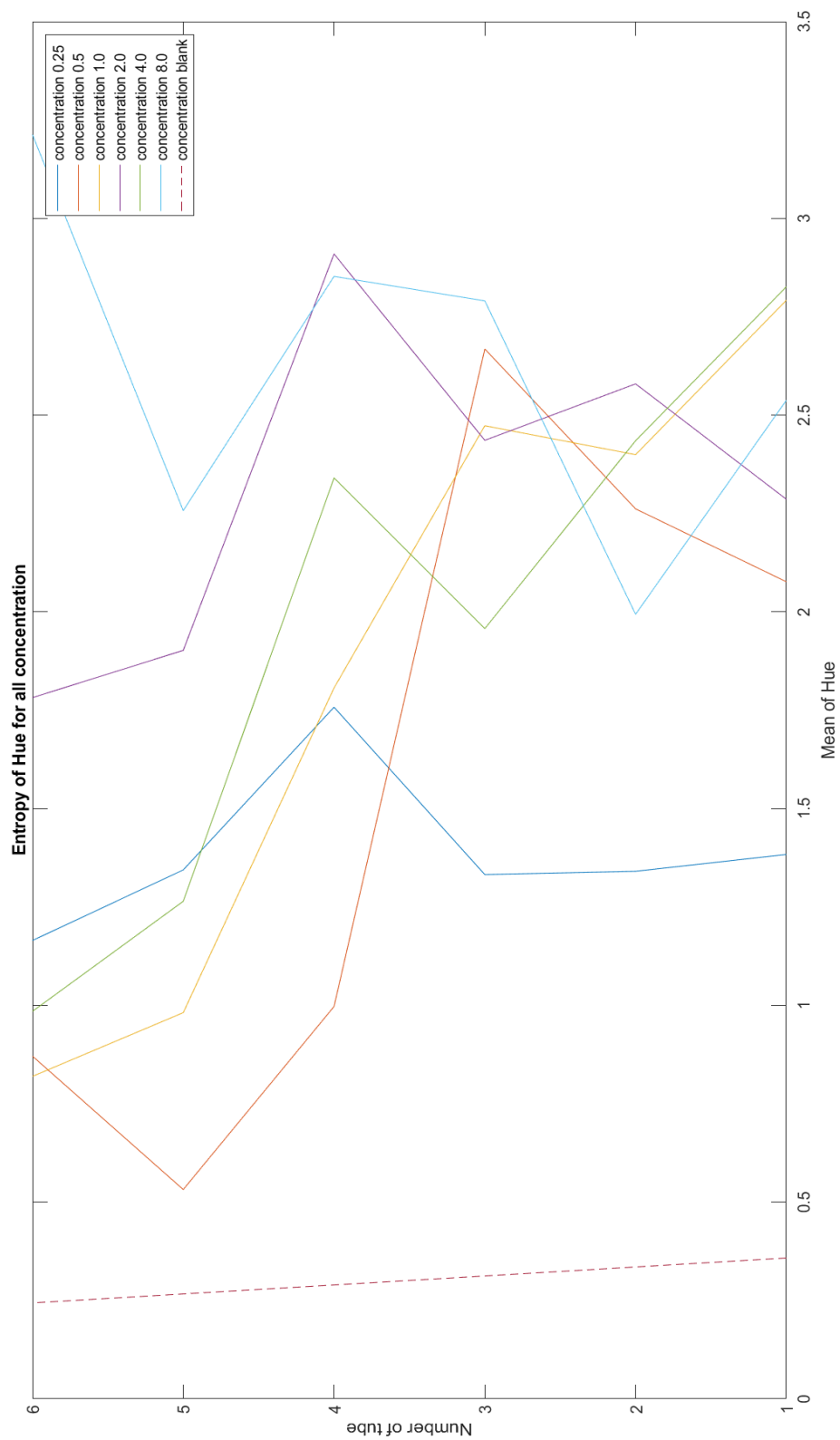


Figure 39: Graph showing the Entropy of Hue for images with different concentration of ammonia.

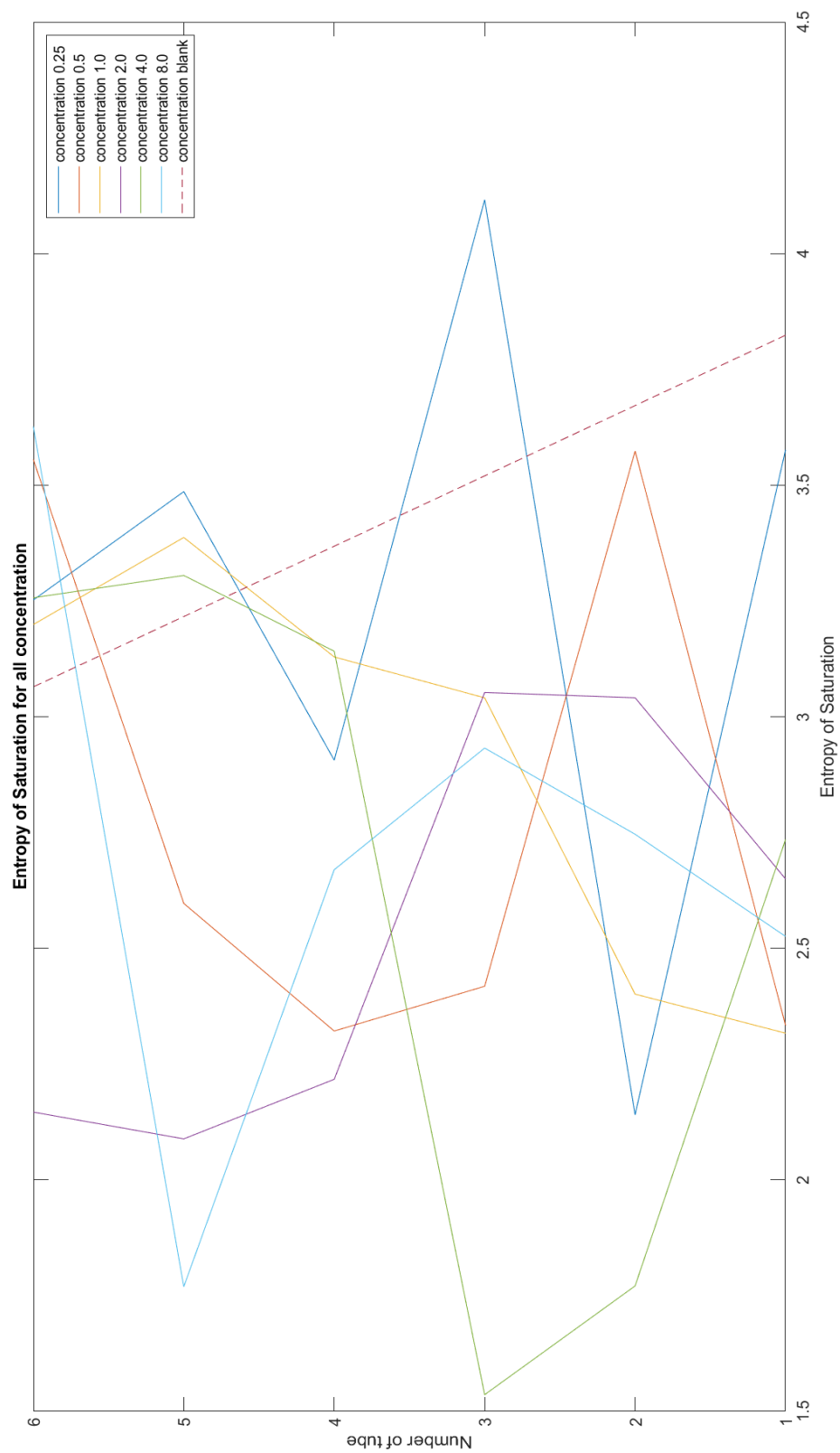


Figure 40: Graph showing the Entropy of Saturation for images with different concentration of ammonia.

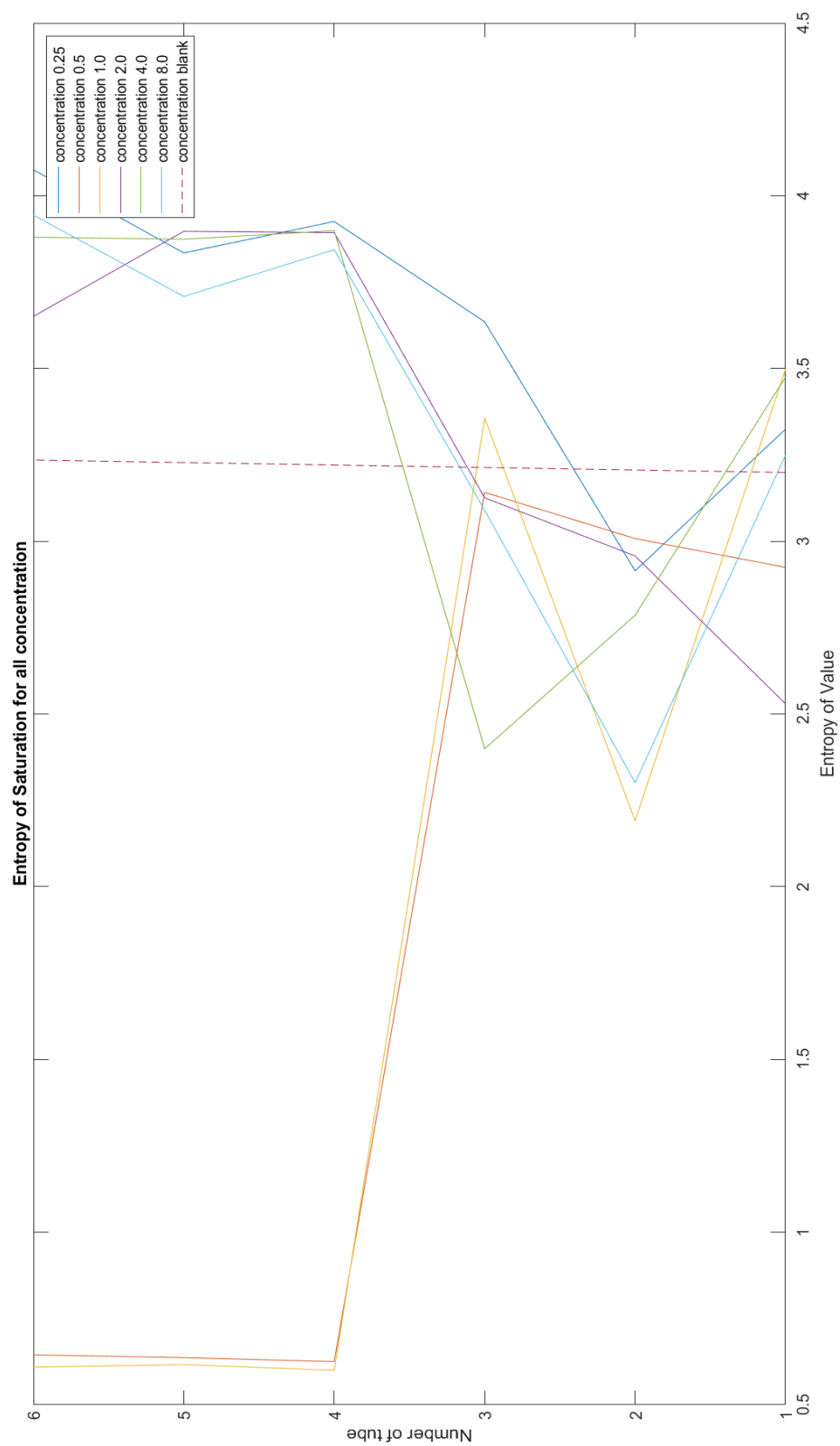


Figure 41: Graph showing the Entropy of Value for images with different concentration of ammonia.



Table 14: Number representation for the concentration

Encoded representation	Concentration
0	Control
1	0.25
2	0.5
3	1
4	2
5	4
6	8

Listing 1.1: K-fold split for training and testing sets for both days

Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 23 24 25 26 27 28 30 31 32 33 34 35 36 37] Validation: [22 29]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37] Validation: [15 16]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 12 13 14 15 16 17 18 19 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37] Validation: [11 20]
Train: [ 0 1 2 3 4 5 6 7 8 9 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 27 28 29 30 31 32 33 34 35 36 37] Validation: [10 26]
Train: [ 0 1 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 26 27 28 29 30 31 32 33 34 35 36 37] Validation: [ 2 25]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 32 34 35 36 37] Validation: [31 33]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 33 34 35 36] Validation: [32 37]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

24 25 26 27 28 29 31 32 33 34 36 37] Validation: [30 35]  
Train: [ 0 1 2 3 4 5 6 7 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24  
25 26 27 28 29 30 31 32 33 35 36 37] Validation: [ 8 34]  
Train: [ 0 1 2 3 4 6 7 8 9 10 11 12 14 15 16 17 18 19 20 21 22 23 24 25  
26 27 28 29 30 31 32 33 34 35 36 37] Validation: [ 5 13]  
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 15 16 18 19 20 21 22 23 24 25  
26 27 28 29 30 31 32 33 34 35 36 37] Validation: [14 17]  
Train: [ 0 1 2 3 4 5 6 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24  
25 26 27 29 30 31 32 33 34 35 36 37] Validation: [ 7 28]  
Train: [ 0 2 3 4 5 6 7 8 9 10 11 13 14 15 16 17 18 19 20 21 22 23 24 25  
26 27 28 29 30 31 32 33 34 35 36 37] Validation: [ 1 12]  
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23  
25 26 28 29 30 31 32 33 34 35 36 37] Validation: [24 27]  
Train: [ 0 1 2 3 4 5 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 24 25  
26 27 28 29 30 31 32 33 34 35 36 37] Validation: [ 6 23]  
Train: [ 0 1 2 3 5 6 7 8 9 10 11 12 13 14 15 16 17 19 20 21 22 23 24 25  
26 27 28 29 30 31 32 33 34 35 36 37] Validation: [ 4 18]  
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 20 22 23 24 25  
26 27 28 29 30 31 32 33 34 35 36 37] Validation: [19 21]  
Train: [ 0 1 2 3 4 5 6 7 8 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24  
25 26 27 28 29 30 31 32 33 34 35 37] Validation: [ 9 36]  
Train: [ 1 2 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25  
26 27 28 29 30 31 32 33 34 35 36 37] Validation: [0 3]  
Random Forest Accuracy: 0.8157894736842105  
Naive Bayes Accuracy: 0.8157894736842105  
Decision Tree Accuracy: 0.8157894736842105

Index	Type	Size	Value
0	float64	1	0.0
1	float64	1	0.5
2	float64	1	0.0
3	float64	1	0.5
4	float64	1	1.0
5	float64	1	0.5
6	float64	1	1.0
7	float64	1	1.0
8	float64	1	1.0
9	float64	1	1.0
10	float64	1	1.0
11	float64	1	1.0
12	float64	1	1.0
13	float64	1	1.0
14	float64	1	1.0
15	float64	1	1.0
16	float64	1	1.0
17	float64	1	1.0
18	float64	1	1.0

Figure 42: Accuracy of prediction (Random forest)

Index	Type	Size	Value
0	float64	1	0.0
1	float64	1	0.5
2	float64	1	0.0
3	float64	1	0.5
4	float64	1	1.0
5	float64	1	0.5
6	float64	1	1.0
7	float64	1	1.0
8	float64	1	1.0
9	float64	1	1.0
10	float64	1	1.0
11	float64	1	1.0
12	float64	1	1.0
13	float64	1	1.0
14	float64	1	1.0
15	float64	1	1.0
16	float64	1	1.0
17	float64	1	1.0
18	float64	1	1.0

Figure 43: Accuracy of prediction (Decision Tree)

Index	Type	Size	Value
0	float64	1	0.0
1	float64	1	0.5
2	float64	1	0.0
3	float64	1	0.5
4	float64	1	1.0
5	float64	1	0.5
6	float64	1	1.0
7	float64	1	1.0
8	float64	1	1.0
9	float64	1	1.0
10	float64	1	1.0
11	float64	1	1.0
12	float64	1	1.0
13	float64	1	1.0
14	float64	1	1.0
15	float64	1	1.0
16	float64	1	1.0
17	float64	1	1.0
18	float64	1	1.0

Figure 44: Accuracy of prediction (Naïve Bayes)

Index	Type	Size	Value
0	int64	(2,)	[1 5]
1	int64	(2,)	[6 5]
2	int64	(2,)	[6 2]
3	int64	(2,)	[6 3]
4	int64	(2,)	[1 3]
5	int64	(2,)	[3 5]
6	int64	(2,)	[5 0]
7	int64	(2,)	[4 6]
8	int64	(2,)	[3 6]
9	int64	(2,)	[2 5]
10	int64	(2,)	[5 6]
11	int64	(2,)	[3 4]
12	int64	(2,)	[1 5]
13	int64	(2,)	[2 3]
14	int64	(2,)	[3 2]
15	int64	(2,)	[2 0]
16	int64	(2,)	[1 1]
17	int64	(2,)	[4 6]
18	int64	(2,)	[1 2]

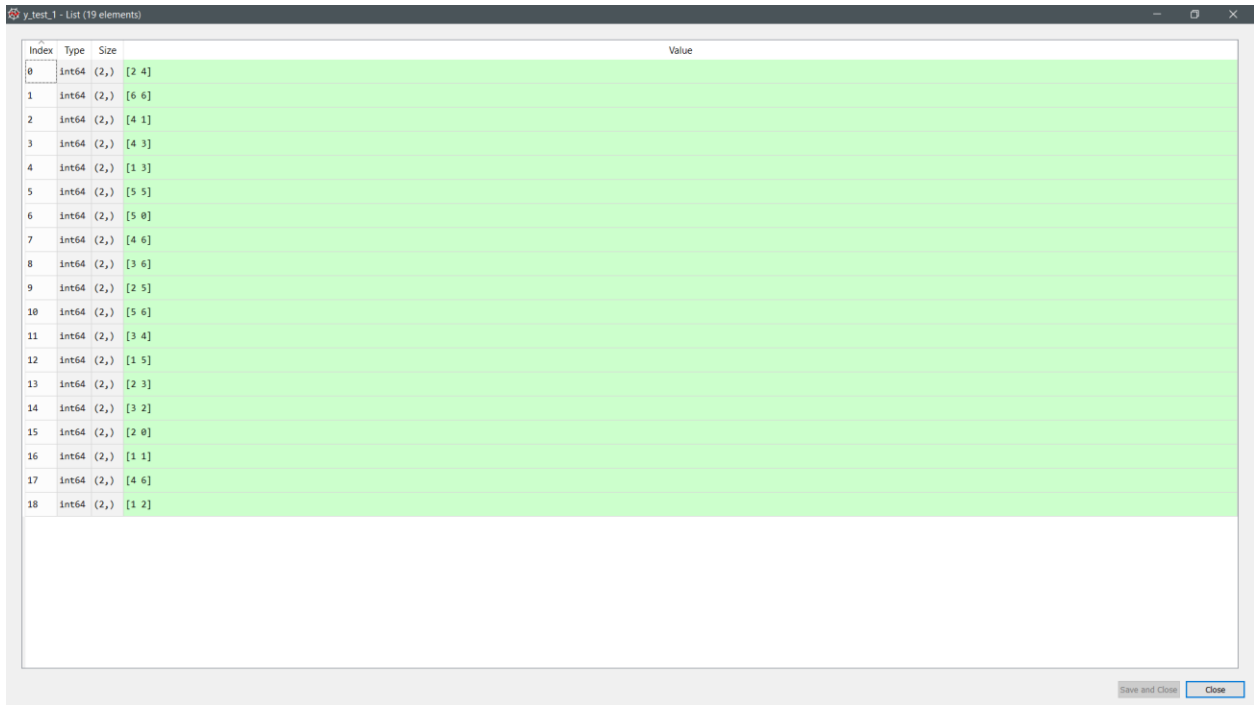
Figure 45: Prediction made by the model (Random forest)

Index	Type	Size	Value
0	int64 (2,)	[1 5]	
1	int64 (2,)	[6 5]	
2	int64 (2,)	[6 2]	
3	int64 (2,)	[6 3]	
4	int64 (2,)	[1 3]	
5	int64 (2,)	[3 5]	
6	int64 (2,)	[5 0]	
7	int64 (2,)	[4 6]	
8	int64 (2,)	[3 6]	
9	int64 (2,)	[2 5]	
10	int64 (2,)	[5 6]	
11	int64 (2,)	[3 4]	
12	int64 (2,)	[1 5]	
13	int64 (2,)	[2 3]	
14	int64 (2,)	[3 2]	
15	int64 (2,)	[2 0]	
16	int64 (2,)	[1 1]	
17	int64 (2,)	[4 6]	
18	int64 (2,)	[1 2]	

Figure 46: Prediction made by the model (Decision Tree)

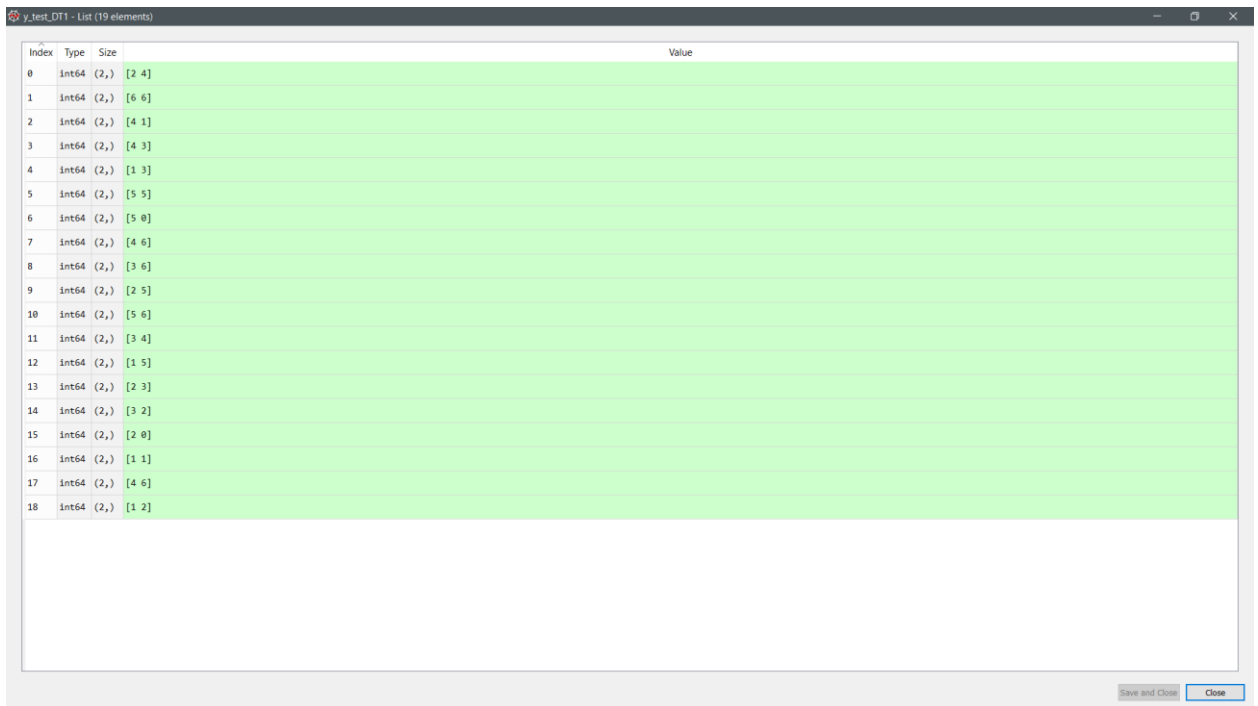
Index	Type	Size	Value
0	int64 (2,)	[2 4]	
1	int64 (2,)	[6 6]	
2	int64 (2,)	[4 1]	
3	int64 (2,)	[4 3]	
4	int64 (2,)	[1 3]	
5	int64 (2,)	[5 5]	
6	int64 (2,)	[5 0]	
7	int64 (2,)	[4 6]	
8	int64 (2,)	[3 6]	
9	int64 (2,)	[2 5]	
10	int64 (2,)	[5 6]	
11	int64 (2,)	[3 4]	
12	int64 (2,)	[1 5]	
13	int64 (2,)	[2 3]	
14	int64 (2,)	[3 2]	
15	int64 (2,)	[2 0]	
16	int64 (2,)	[1 1]	
17	int64 (2,)	[4 6]	
18	int64 (2,)	[1 2]	

Figure 47: Prediction made by the model (Naïve Bayes)



Index	Type	Size	Value
0	int64	(2,)	[2 4]
1	int64	(2,)	[6 6]
2	int64	(2,)	[4 1]
3	int64	(2,)	[4 3]
4	int64	(2,)	[1 3]
5	int64	(2,)	[5 5]
6	int64	(2,)	[5 0]
7	int64	(2,)	[4 6]
8	int64	(2,)	[3 6]
9	int64	(2,)	[2 5]
10	int64	(2,)	[5 6]
11	int64	(2,)	[3 4]
12	int64	(2,)	[1 5]
13	int64	(2,)	[2 3]
14	int64	(2,)	[3 2]
15	int64	(2,)	[2 0]
16	int64	(2,)	[1 1]
17	int64	(2,)	[4 6]
18	int64	(2,)	[1 2]

Figure 48: Actual value (Random Forest)



Index	Type	Size	Value
0	int64	(2,)	[2 4]
1	int64	(2,)	[6 6]
2	int64	(2,)	[4 1]
3	int64	(2,)	[4 3]
4	int64	(2,)	[1 3]
5	int64	(2,)	[5 5]
6	int64	(2,)	[5 0]
7	int64	(2,)	[4 6]
8	int64	(2,)	[3 6]
9	int64	(2,)	[2 5]
10	int64	(2,)	[5 6]
11	int64	(2,)	[3 4]
12	int64	(2,)	[1 5]
13	int64	(2,)	[2 3]
14	int64	(2,)	[3 2]
15	int64	(2,)	[2 0]
16	int64	(2,)	[1 1]
17	int64	(2,)	[4 6]
18	int64	(2,)	[1 2]

Figure 49: Actual value (Decision Tree)

y_test_NB1 - List (19 elements)			
Index	Type	Size	Value
0	int64	(2,)	[2 4]
1	int64	(2,)	[6 6]
2	int64	(2,)	[4 1]
3	int64	(2,)	[4 3]
4	int64	(2,)	[1 3]
5	int64	(2,)	[5 5]
6	int64	(2,)	[5 0]
7	int64	(2,)	[4 6]
8	int64	(2,)	[3 6]
9	int64	(2,)	[2 5]
10	int64	(2,)	[5 6]
11	int64	(2,)	[3 4]
12	int64	(2,)	[1 5]
13	int64	(2,)	[2 3]
14	int64	(2,)	[3 2]
15	int64	(2,)	[2 0]
16	int64	(2,)	[1 1]
17	int64	(2,)	[4 6]
18	int64	(2,)	[1 2]

Figure 50: Actual value (Naïve Bayes)

Listing 1.2: K-fold split for training and testing sets for day 1 only

```
Train: [ 0 1 2 3 4 5 6 7 8 9 11 12 13 14 15 16 17 18] Validation: [10]
Train: [ 0 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18] Validation: [1]
Train: [ 0 1 2 3 4 5 6 7 9 10 11 12 13 14 15 16 17 18] Validation: [8]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17] Validation: [18]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 15 16 17 18] Validation: [14]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 17 18] Validation: [16]
Train: [ 0 1 2 3 4 5 7 8 9 10 11 12 13 14 15 16 17 18] Validation: [6]
Train: [ 0 1 2 3 5 6 7 8 9 10 11 12 13 14 15 16 17 18] Validation: [4]
Train: [ 0 1 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18] Validation: [2]
Train: [ 0 1 2 3 4 6 7 8 9 10 11 12 13 14 15 16 17 18] Validation: [5]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 14 15 16 17 18] Validation: [13]
Train: [ 0 1 2 3 4 5 6 7 8 10 11 12 13 14 15 16 17 18] Validation: [9]
Train: [ 0 1 2 3 4 5 6 8 9 10 11 12 13 14 15 16 17 18] Validation: [7]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 18] Validation: [17]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 12 13 14 15 16 17 18] Validation: [11]
Train: [ 0 1 2 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18] Validation: [3]
Train: [ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18] Validation: [0]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 16 17 18] Validation: [15]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 13 14 15 16 17 18] Validation: [12]

Random Forest Accuracy: 0.9473684210526315
Naïve Bayes Accuracy: 0.9473684210526315
Decision Tree Accuracy: 0.9473684210526315
```

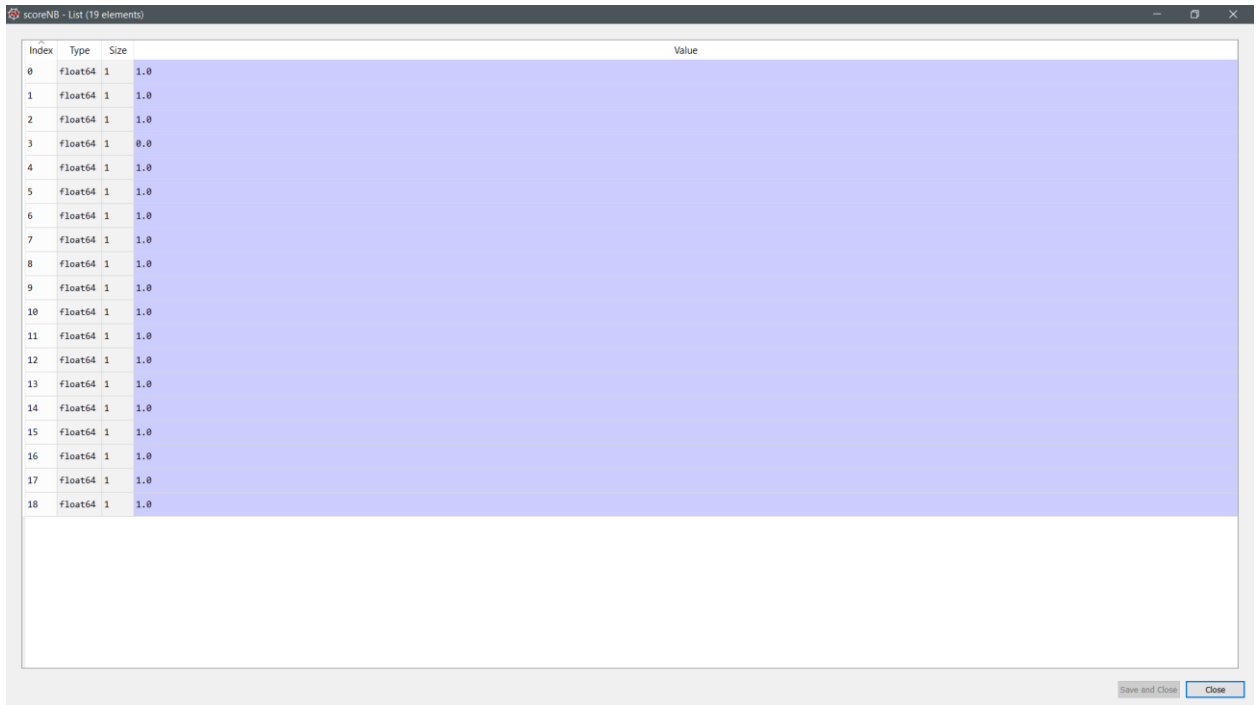


Index	Type	Size	Value
0	float64	1	1.0
1	float64	1	1.0
2	float64	1	1.0
3	float64	1	0.0
4	float64	1	1.0
5	float64	1	1.0
6	float64	1	1.0
7	float64	1	1.0
8	float64	1	1.0
9	float64	1	1.0
10	float64	1	1.0
11	float64	1	1.0
12	float64	1	1.0
13	float64	1	1.0
14	float64	1	1.0
15	float64	1	1.0
16	float64	1	1.0
17	float64	1	1.0
18	float64	1	1.0

Figure 51: Accuracy of prediction (Random forest)

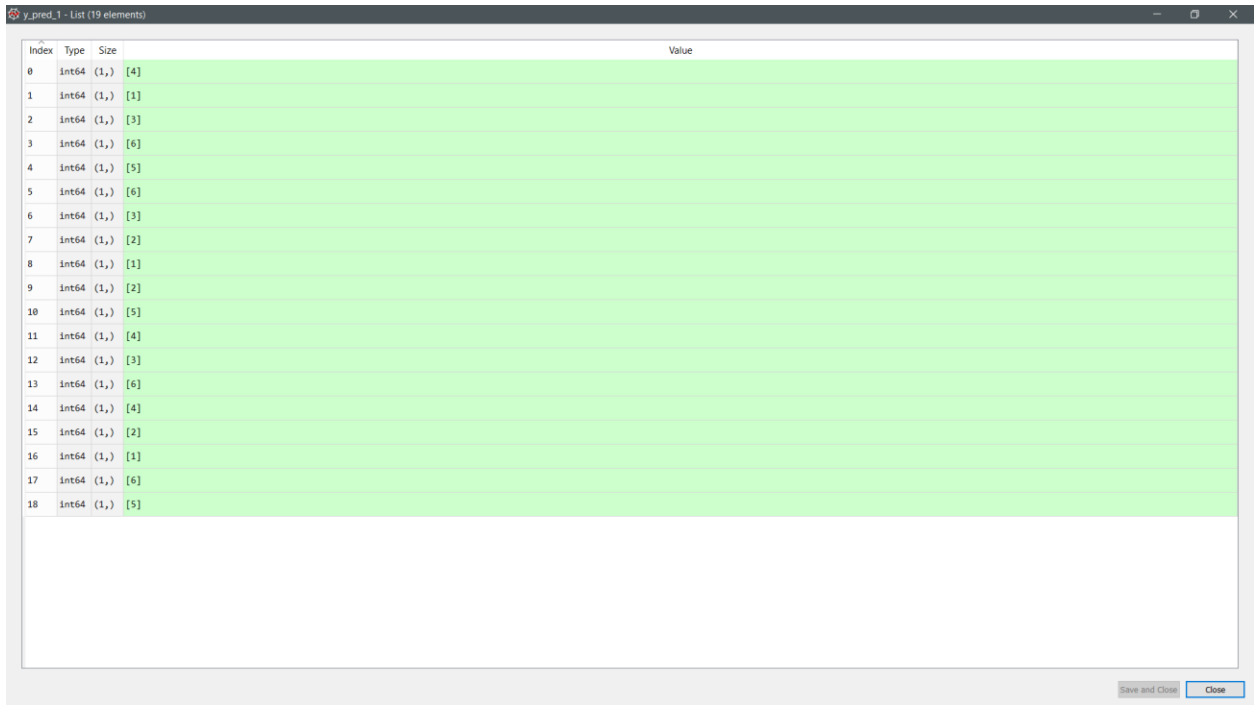
Index	Type	Size	Value
0	float64	1	1.0
1	float64	1	1.0
2	float64	1	1.0
3	float64	1	0.0
4	float64	1	1.0
5	float64	1	1.0
6	float64	1	1.0
7	float64	1	1.0
8	float64	1	1.0
9	float64	1	1.0
10	float64	1	1.0
11	float64	1	1.0
12	float64	1	1.0
13	float64	1	1.0
14	float64	1	1.0
15	float64	1	1.0
16	float64	1	1.0
17	float64	1	1.0
18	float64	1	1.0

Figure 52: Accuracy of prediction (Decision Tree)



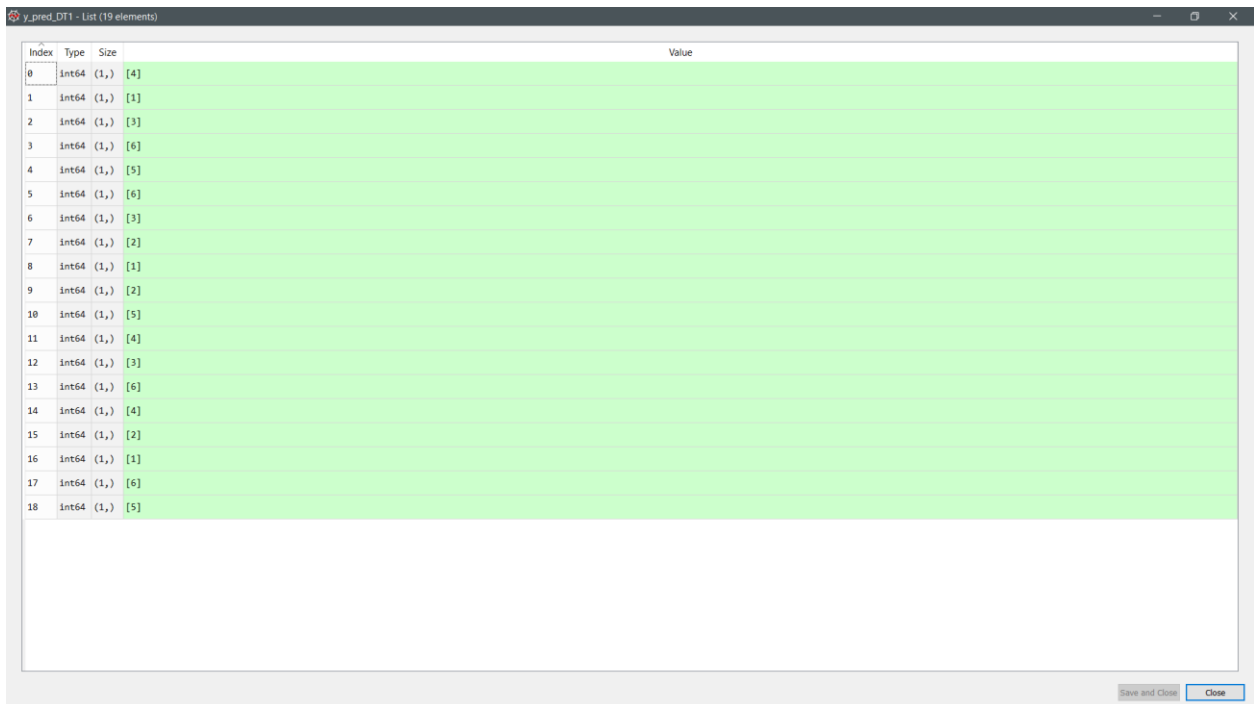
Index	Type	Size	Value
0	float64	1	1.0
1	float64	1	1.0
2	float64	1	1.0
3	float64	1	0.0
4	float64	1	1.0
5	float64	1	1.0
6	float64	1	1.0
7	float64	1	1.0
8	float64	1	1.0
9	float64	1	1.0
10	float64	1	1.0
11	float64	1	1.0
12	float64	1	1.0
13	float64	1	1.0
14	float64	1	1.0
15	float64	1	1.0
16	float64	1	1.0
17	float64	1	1.0
18	float64	1	1.0

Figure 53: Accuracy of prediction (Naïve Bayes)



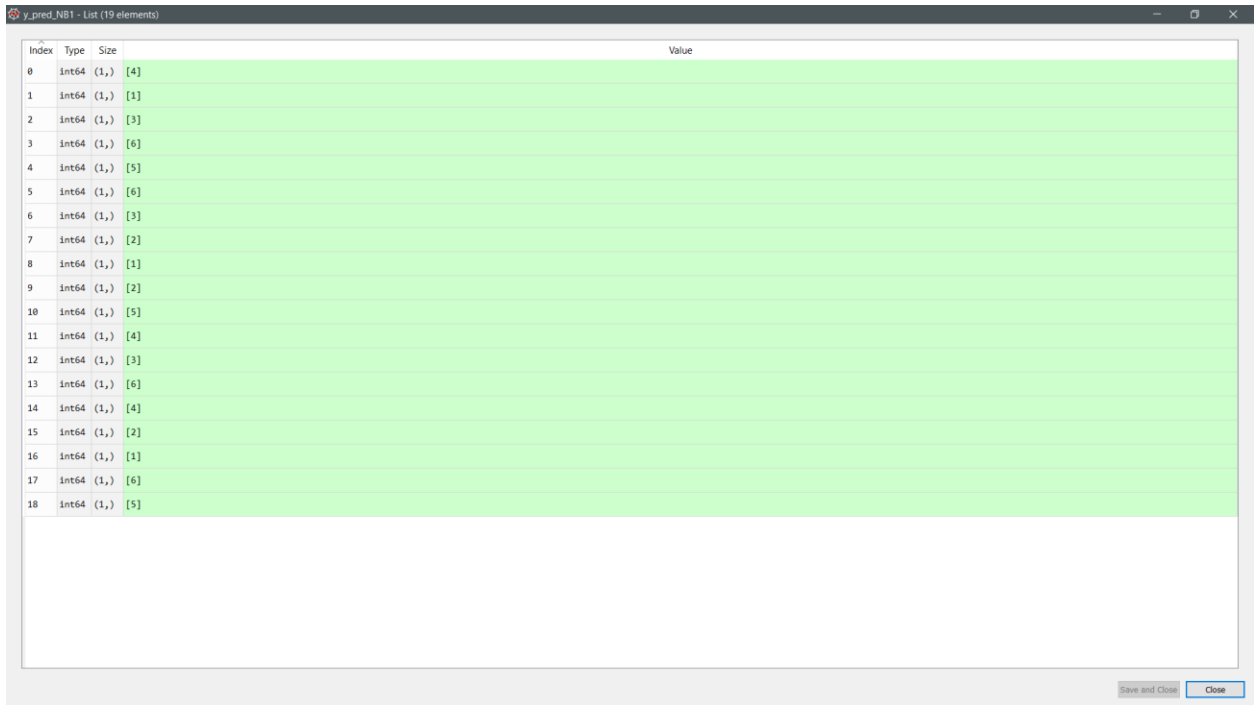
Index	Type	Size	Value
0	int64	(1,)	[4]
1	int64	(1,)	[1]
2	int64	(1,)	[3]
3	int64	(1,)	[6]
4	int64	(1,)	[5]
5	int64	(1,)	[6]
6	int64	(1,)	[3]
7	int64	(1,)	[2]
8	int64	(1,)	[1]
9	int64	(1,)	[2]
10	int64	(1,)	[5]
11	int64	(1,)	[4]
12	int64	(1,)	[3]
13	int64	(1,)	[6]
14	int64	(1,)	[4]
15	int64	(1,)	[2]
16	int64	(1,)	[1]
17	int64	(1,)	[6]
18	int64	(1,)	[5]

Figure 54: Prediction made by the model (Random forest)



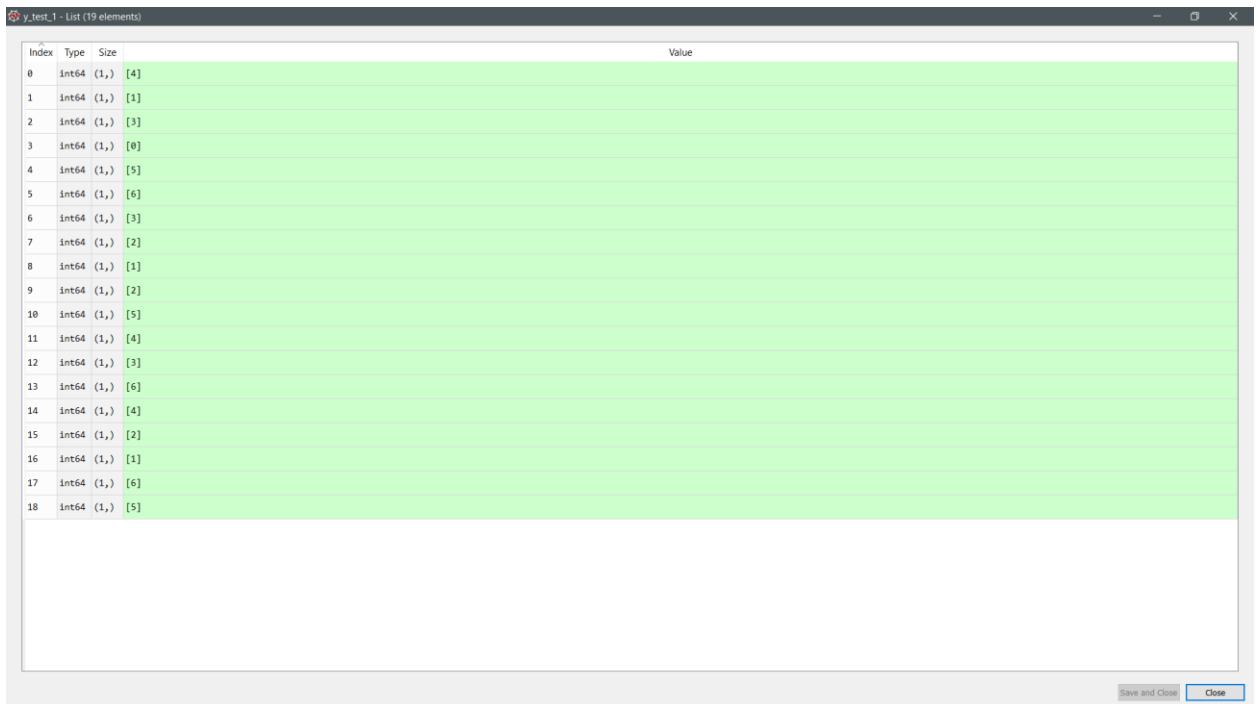
Index	Type	Size	Value
0	int64	(1,)	[4]
1	int64	(1,)	[1]
2	int64	(1,)	[3]
3	int64	(1,)	[6]
4	int64	(1,)	[5]
5	int64	(1,)	[6]
6	int64	(1,)	[3]
7	int64	(1,)	[2]
8	int64	(1,)	[1]
9	int64	(1,)	[2]
10	int64	(1,)	[5]
11	int64	(1,)	[4]
12	int64	(1,)	[3]
13	int64	(1,)	[6]
14	int64	(1,)	[4]
15	int64	(1,)	[2]
16	int64	(1,)	[1]
17	int64	(1,)	[6]
18	int64	(1,)	[5]

Figure 55: Prediction made by the model (Decision Tree)



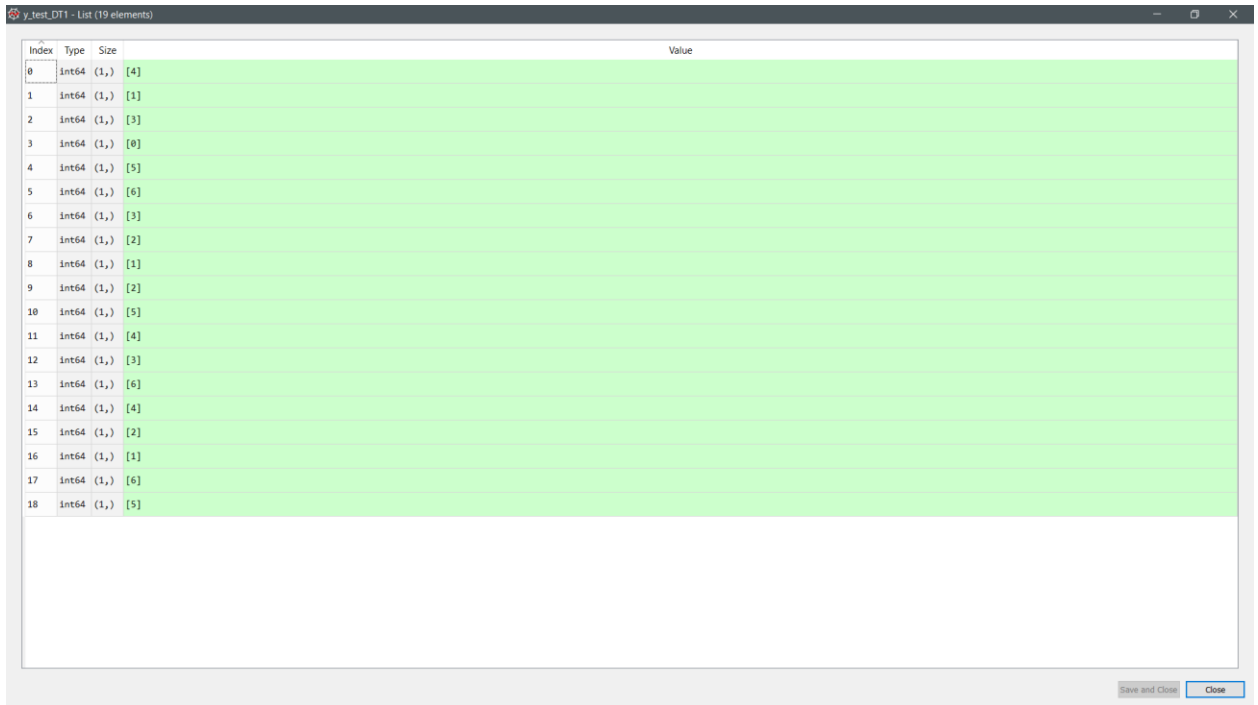
Index	Type	Size	Value
0	int64	(1,)	[4]
1	int64	(1,)	[1]
2	int64	(1,)	[3]
3	int64	(1,)	[6]
4	int64	(1,)	[5]
5	int64	(1,)	[6]
6	int64	(1,)	[3]
7	int64	(1,)	[2]
8	int64	(1,)	[1]
9	int64	(1,)	[2]
10	int64	(1,)	[5]
11	int64	(1,)	[4]
12	int64	(1,)	[3]
13	int64	(1,)	[6]
14	int64	(1,)	[4]
15	int64	(1,)	[2]
16	int64	(1,)	[1]
17	int64	(1,)	[6]
18	int64	(1,)	[5]

Figure 56: Prediction made by the model (Naive Bayes)



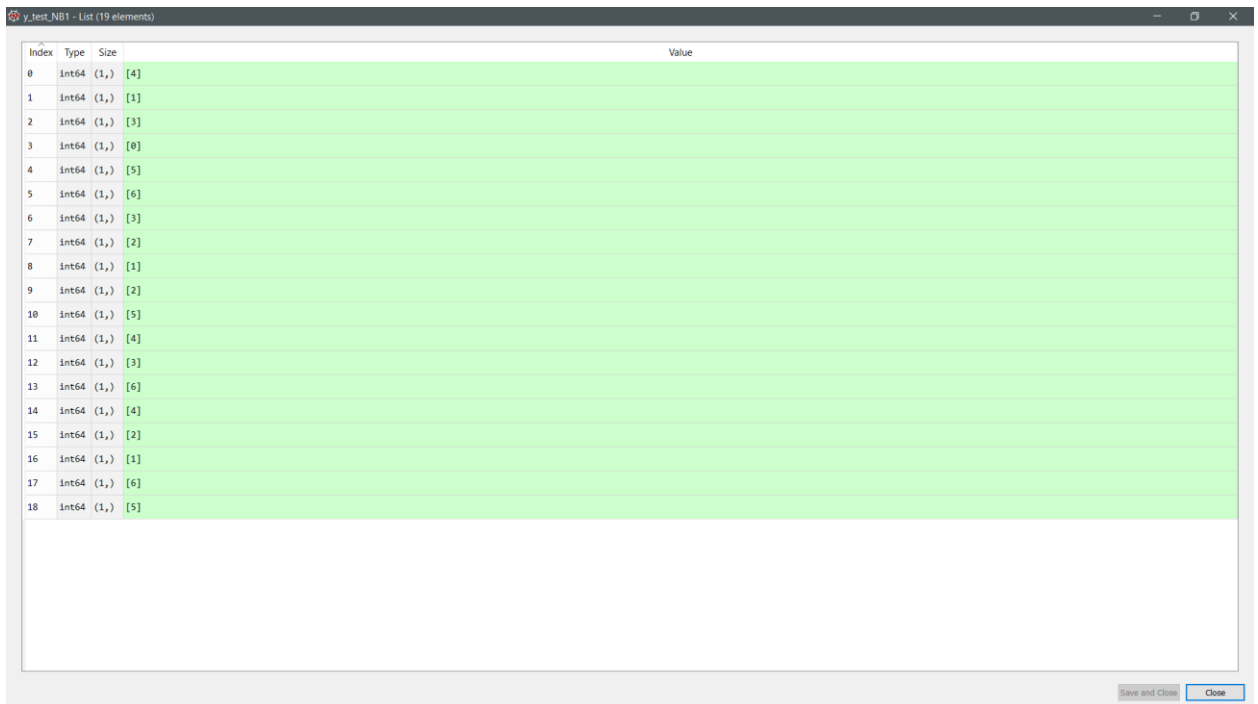
Index	Type	Size	Value
0	int64	(1,)	[4]
1	int64	(1,)	[1]
2	int64	(1,)	[3]
3	int64	(1,)	[0]
4	int64	(1,)	[5]
5	int64	(1,)	[6]
6	int64	(1,)	[3]
7	int64	(1,)	[2]
8	int64	(1,)	[1]
9	int64	(1,)	[2]
10	int64	(1,)	[5]
11	int64	(1,)	[4]
12	int64	(1,)	[3]
13	int64	(1,)	[6]
14	int64	(1,)	[4]
15	int64	(1,)	[2]
16	int64	(1,)	[1]
17	int64	(1,)	[6]
18	int64	(1,)	[5]

Figure 57: Actual value (Random Forest)



Index	Type	Size	Value
0	int64	(1,)	[4]
1	int64	(1,)	[1]
2	int64	(1,)	[3]
3	int64	(1,)	[0]
4	int64	(1,)	[5]
5	int64	(1,)	[6]
6	int64	(1,)	[3]
7	int64	(1,)	[2]
8	int64	(1,)	[1]
9	int64	(1,)	[2]
10	int64	(1,)	[5]
11	int64	(1,)	[4]
12	int64	(1,)	[3]
13	int64	(1,)	[6]
14	int64	(1,)	[4]
15	int64	(1,)	[2]
16	int64	(1,)	[1]
17	int64	(1,)	[6]
18	int64	(1,)	[5]

Figure 58: Actual value (Decision Tree)



Index	Type	Size	Value
0	int64	(1,)	[4]
1	int64	(1,)	[1]
2	int64	(1,)	[3]
3	int64	(1,)	[0]
4	int64	(1,)	[5]
5	int64	(1,)	[6]
6	int64	(1,)	[3]
7	int64	(1,)	[2]
8	int64	(1,)	[1]
9	int64	(1,)	[2]
10	int64	(1,)	[5]
11	int64	(1,)	[4]
12	int64	(1,)	[3]
13	int64	(1,)	[6]
14	int64	(1,)	[4]
15	int64	(1,)	[2]
16	int64	(1,)	[1]
17	int64	(1,)	[6]
18	int64	(1,)	[5]

Figure 59: Actual value (Naïve Bayes)

Listing 1.3: K-fold split for training and testing sets for day 2 only

Train: [ 0 1 2 3 4 5 6 7 8 9 11 12 13 14 15 16 17 18] Validation: [10]
Train: [ 0 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18] Validation: [1]
Train: [ 0 1 2 3 4 5 6 7 9 10 11 12 13 14 15 16 17 18] Validation: [8]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17] Validation: [18]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 15 16 17 18] Validation: [14]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 17 18] Validation: [16]
Train: [ 0 1 2 3 4 5 7 8 9 10 11 12 13 14 15 16 17 18] Validation: [6]
Train: [ 0 1 2 3 5 6 7 8 9 10 11 12 13 14 15 16 17 18] Validation: [4]
Train: [ 0 1 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18] Validation: [2]
Train: [ 0 1 2 3 4 6 7 8 9 10 11 12 13 14 15 16 17 18] Validation: [5]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 14 15 16 17 18] Validation: [13]
Train: [ 0 1 2 3 4 5 6 7 8 10 11 12 13 14 15 16 17 18] Validation: [9]
Train: [ 0 1 2 3 4 5 6 8 9 10 11 12 13 14 15 16 17 18] Validation: [7]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 18] Validation: [17]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 12 13 14 15 16 17 18] Validation: [11]
Train: [ 0 1 2 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18] Validation: [3]
Train: [ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18] Validation: [0]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 16 17 18] Validation: [15]
Train: [ 0 1 2 3 4 5 6 7 8 9 10 11 13 14 15 16 17 18] Validation: [12]
Random Forest Accuracy: 0.9473684210526315
Naïve Bayes Accuracy: 0.9473684210526315
Decision Tree Accuracy: 0.9473684210526315

Index	Type	Size	Value
0	float64	1	1.0
1	float64	1	1.0
2	float64	1	1.0
3	float64	1	0.0
4	float64	1	1.0
5	float64	1	1.0
6	float64	1	1.0
7	float64	1	1.0
8	float64	1	1.0
9	float64	1	1.0
10	float64	1	1.0
11	float64	1	1.0
12	float64	1	1.0
13	float64	1	1.0
14	float64	1	1.0
15	float64	1	1.0
16	float64	1	1.0
17	float64	1	1.0
18	float64	1	1.0

Figure 60: Accuracy of prediction (Random Forest)

Index	Type	Size	Value
0	float64	1	1.0
1	float64	1	1.0
2	float64	1	1.0
3	float64	1	0.0
4	float64	1	1.0
5	float64	1	1.0
6	float64	1	1.0
7	float64	1	1.0
8	float64	1	1.0
9	float64	1	1.0
10	float64	1	1.0
11	float64	1	1.0
12	float64	1	1.0
13	float64	1	1.0
14	float64	1	1.0
15	float64	1	1.0
16	float64	1	1.0
17	float64	1	1.0
18	float64	1	1.0

Figure 61: Accuracy of prediction (Decision Tree)

Index	Type	Size	Value
0	float64	1	1.0
1	float64	1	1.0
2	float64	1	1.0
3	float64	1	0.0
4	float64	1	1.0
5	float64	1	1.0
6	float64	1	1.0
7	float64	1	1.0
8	float64	1	1.0
9	float64	1	1.0
10	float64	1	1.0
11	float64	1	1.0
12	float64	1	1.0
13	float64	1	1.0
14	float64	1	1.0
15	float64	1	1.0
16	float64	1	1.0
17	float64	1	1.0
18	float64	1	1.0

Figure 62: Accuracy of prediction (Naïve Bayes)

Index	Type	Size	Value
0	int64 (1,)	[4]	
1	int64 (1,)	[1]	
2	int64 (1,)	[3]	
3	int64 (1,)	[6]	
4	int64 (1,)	[5]	
5	int64 (1,)	[6]	
6	int64 (1,)	[3]	
7	int64 (1,)	[2]	
8	int64 (1,)	[1]	
9	int64 (1,)	[2]	
10	int64 (1,)	[5]	
11	int64 (1,)	[4]	
12	int64 (1,)	[3]	
13	int64 (1,)	[6]	
14	int64 (1,)	[4]	
15	int64 (1,)	[2]	
16	int64 (1,)	[1]	
17	int64 (1,)	[6]	
18	int64 (1,)	[5]	

Figure 63: Prediction made by the model (Random Forest)



Index	Type	Size	Value
0	int64	(1,)	[4]
1	int64	(1,)	[1]
2	int64	(1,)	[3]
3	int64	(1,)	[6]
4	int64	(1,)	[5]
5	int64	(1,)	[6]
6	int64	(1,)	[3]
7	int64	(1,)	[2]
8	int64	(1,)	[1]
9	int64	(1,)	[2]
10	int64	(1,)	[5]
11	int64	(1,)	[4]
12	int64	(1,)	[3]
13	int64	(1,)	[6]
14	int64	(1,)	[4]
15	int64	(1,)	[2]
16	int64	(1,)	[1]
17	int64	(1,)	[6]
18	int64	(1,)	[5]

Figure 64: Prediction made by the model (Decision Tree)

Index	Type	Size	Value
0	int64	(1,)	[4]
1	int64	(1,)	[1]
2	int64	(1,)	[3]
3	int64	(1,)	[6]
4	int64	(1,)	[5]
5	int64	(1,)	[6]
6	int64	(1,)	[3]
7	int64	(1,)	[2]
8	int64	(1,)	[1]
9	int64	(1,)	[2]
10	int64	(1,)	[5]
11	int64	(1,)	[4]
12	int64	(1,)	[3]
13	int64	(1,)	[6]
14	int64	(1,)	[4]
15	int64	(1,)	[2]
16	int64	(1,)	[1]
17	int64	(1,)	[6]
18	int64	(1,)	[5]

Figure 65: Prediction made by the model (Naive Bayes)

Index	Type	Size	Value
0	int64	(1,)	[4]
1	int64	(1,)	[1]
2	int64	(1,)	[3]
3	int64	(1,)	[0]
4	int64	(1,)	[5]
5	int64	(1,)	[6]
6	int64	(1,)	[3]
7	int64	(1,)	[2]
8	int64	(1,)	[1]
9	int64	(1,)	[2]
10	int64	(1,)	[5]
11	int64	(1,)	[4]
12	int64	(1,)	[3]
13	int64	(1,)	[6]
14	int64	(1,)	[4]
15	int64	(1,)	[2]
16	int64	(1,)	[1]
17	int64	(1,)	[6]
18	int64	(1,)	[5]

Figure 66: Actual value (Random Forest)

Index	Type	Size	Value
0	int64	(1,)	[4]
1	int64	(1,)	[1]
2	int64	(1,)	[3]
3	int64	(1,)	[0]
4	int64	(1,)	[5]
5	int64	(1,)	[6]
6	int64	(1,)	[3]
7	int64	(1,)	[2]
8	int64	(1,)	[1]
9	int64	(1,)	[2]
10	int64	(1,)	[5]
11	int64	(1,)	[4]
12	int64	(1,)	[3]
13	int64	(1,)	[6]
14	int64	(1,)	[4]
15	int64	(1,)	[2]
16	int64	(1,)	[1]
17	int64	(1,)	[6]
18	int64	(1,)	[5]

Figure 67: Actual value (Decision Tree)

y_test_NB1 - List (19 elements)			
Index	Type	Size	Value
0	int64	(1,)	[4]
1	int64	(1,)	[1]
2	int64	(1,)	[3]
3	int64	(1,)	[0]
4	int64	(1,)	[5]
5	int64	(1,)	[6]
6	int64	(1,)	[3]
7	int64	(1,)	[2]
8	int64	(1,)	[1]
9	int64	(1,)	[2]
10	int64	(1,)	[5]
11	int64	(1,)	[4]
12	int64	(1,)	[3]
13	int64	(1,)	[6]
14	int64	(1,)	[4]
15	int64	(1,)	[2]
16	int64	(1,)	[1]
17	int64	(1,)	[6]
18	int64	(1,)	[5]

Figure 68: Actual value (Naïve Bayes)

## 6.0 Discussion

First of all, the Region of Interest (ROI) for this assignment is the water in the test tube. However, the sample images contain information that is unwanted such as the background and the label of the test tube. Therefore, the result will be inaccurate if no pre-processing is done. To simplify the processing, the position of test tube captured in the image is assumed constant. By using the crop function “`imcrop()`,” in MATLAB, the ROI can be extracted by setting the position manually. The background is completely removed after cropping. Still, the images have undesired information, specifically, the labels on the test tubes. Therefore, a 2-D median filter MATLAB function “`medfilt2( )`,” is applied to remove that particular information. The median filter replaces each pixel value with the median of its neighboring pixel values. (Robert and Simon, 2004) In this case, the size of the median filter mask is 200×200 pixels in order to obtain a desired result. Since the median for each point in the image may be slightly different, the resulting images will have shading. To reduce the shading effect, an average filter is defined by “`fspecial(‘average’, [ ])`,” and applied through “`imfilter( )`,”. This operation will smoothen the images by considering the average in the neighborhood for each of the pixels in the images. Finally, the images are ready for measurement of features such as Mean and Entropy of their RGB and HSV model.

RGB color model is one of the simplest and widely used color representation method in image processing. Also, HSV model is used as it is closer to how humans perceive colors. The three components in HSV model are Hue, Saturation, and Value. (Jacci, 2019) Basically, Hue represents color as shown in Table 15, Saturation depicts the range of grey in color space as illustrated in Figure 69, and Value describes the brightness or intensity of the color. (Pooja, 2016) The HSV color model of the images are obtained from their respective RGB color model through MATLAB function, “`rgb2hsv( )`,”. Each of the elements in the color models are stored as matrix variables separately. These variables are required for the determination of the parameters needed for machine learning.

Table 15: Table showing the corresponding color for the ranges of Hue.

Angle / °	Color
0 - 60	Red
61 - 120	Yellow
121 - 180	Green
181 - 240	Cyan
241 - 300	Blue
301 - 360	Magenta

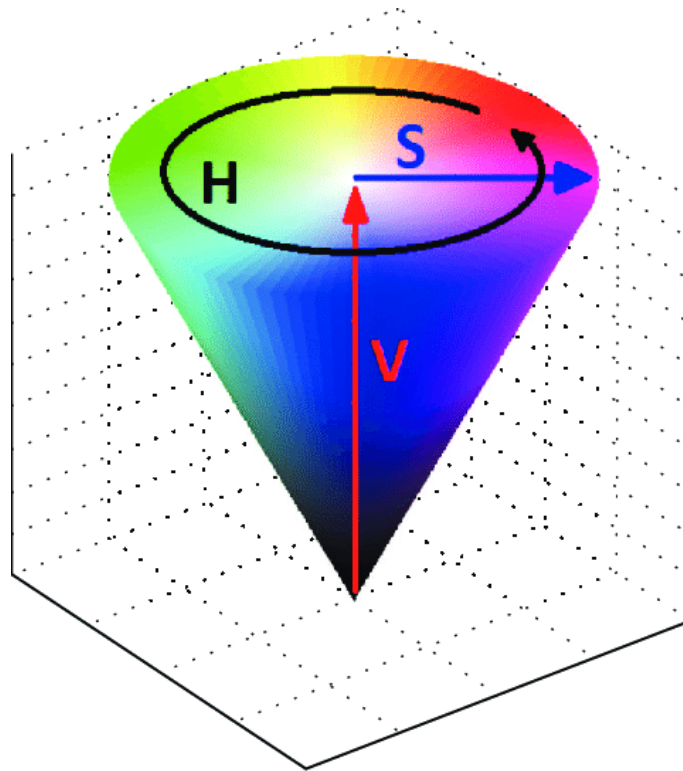


Figure 69: The HSV color model where H = Hue, S = Saturation, and V = Value.

To determine the Mean and Entropy, MATLAB functions “mean2( );” and “entropy( );” are applied respectively to each of the elements in the color models. Mean is the average of all the intensities of the pixels in an image. Since the filtered images are smooth images, mean can be a reliable parameter for the prediction of the color. Notice in Figure 30 to 35, the mean curve is overlapping in multiple ammonia concentration. Standard deviation might be a good choice to counter the issue as it tells us how much does the mean spread out in a data set. However, it is not effective when it comes to machine learning since standard deviation itself is a function of mean. In other words, standard deviation does not give any informative features that can be used for the prediction. On the other hand, Entropy is a statistical measure of randomness of gray levels in an image. It is used to characterize the texture of an image. (Rafael, Richard and Steven, 2003) It can be expressed as  $E = - \sum_{i=0}^{2B-1} p_i \log_2 p_i$ , where  $p_i$  is the probability of occurrence of color  $i$  and  $B$  is the total number of bits in the digitized image. Notice that Entropy does not only depend on  $p_i$ . In fact, it also depends on the quantity of gray levels of the image. Therefore, Entropy is a significant feature needed to be analyzed, especially for comparison of images with different quantity of gray levels. (Esley, Yasel, Osvaldo and Roberto, 2015)

After those images were processed in the MATLAB, the mean and entropy of the processed images for both days were stored in an excel file. Three machine learning algorithms were implemented to classify the data according to their concentration. The machine learning algorithms that were used are random forest, decision tree and Naïve Bayes. Gupta (2017) states that a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. While a random forest is a collection of decision trees whose results are aggregated into one final result. Random forest is a stronger predictive model compared to decision tree, because it solve the problem of overfitting by minimizing both error due to bias and variance.

A Naive Bayes classifier is a probabilistic machine learning model that's used for classification task. The crux of the classifier is based on the Bayes theorem. Using Bayes theorem, we can find the probability of **A** happening, given that **B** has occurred. Here, **B** is

the evidence and  $\mathbf{A}$  is the hypothesis. The assumption made here is that the predictors/features are independent. That is presence of one particular feature does not affect the other. Hence it is called naive Gandhi (2018). The probability formula for Bayes theorem was shown as below:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (6.1)$$

The Naïve Bayes classifier that was used in this project is Gaussian Naïve Bayes. Gandhi (2018) states that when the predictors take up a continuous value and are not discrete, we assume that these values are sampled from a gaussian distribution. Since the way the values are present in the dataset changes, the formula for conditional probability changes to,

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} \exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right) \quad (6.2)$$

Table 14 shows the encoded label representation of the concentration for the ammonia test kit. The label for the classifier must be an integer, hence the concentration was encoded in order to conduct prediction. K-fold cross validation was implemented to separate the dataset into two, which are train and test dataset. Hewa (2018) states that K-fold Cross Validation (CV) divides the data into folds and ensuring that each fold is used as a testing set at some point. The k value was set to 19, which means that there are 19 folds, for each fold there are 1 testing set and 18 training sets. The features testing and training sets were normalized before putting it into the training model.

Next, the model was trained with the dataset that was split by the K-fold cross validation method. A testing set was fed into the trained model to get the predicted value and a confusion matrix was generated using the predicted value. From Figure 42 to 44 shows the accuracy of the prediction models, while Figure 45 to 47 shows the predictions of three of the models and Figure 48 to 50 shows the actual value of the data.

The accuracy for all three of the prediction models are the same, which is 81.58% for the dataset of both days, the accuracy for three of the models was shown in Listing 1.1. Moreover, for the dataset of day 1 only, the accuracy for all three of the models are the

same too, comes with the accuracy of 94.74% which was shown in Listing 1.2. Lastly, for the dataset of day 2 only, the accuracy for all three of the models are still the same, with the accuracy of 94.74% which is shown in Listing 1.3.

Moreover, when both days of the results were combined together to train three of the classifier models, the colour for the control concentration can be predicted successfully. The control concentration was placed at the 7<sup>th</sup> and 16<sup>th</sup> fold (6<sup>th</sup> and 15<sup>th</sup> elements) from Figure 42 to Figure 50. From Figure 42 to Figure 44, the accuracy for the prediction of the control concentration is 100%. Continuing Figure 45 to Figure 47 shows the prediction for the testing dataset of the control concentration, by comparing it to Figure 48 to Figure 50, the prediction of the trained models is the same as the actual value. However, while training those models for the 2 days separately, those models was unable to predict the concentration for the control successfully. This is because there is only one data value on the control for both days respectively. But when training both days of data values together, the trained models was able to predict the control concentration successfully because there are two control concentration data values in the dataset.

For a bigger dataset, the accuracy of Random Forest will be the highest, followed by Naïve Bayes, then the last will be Decision Tree. Since Random Forest is a collection of Decision Trees, and which it was developed to overcome the overfitting problem, high bias and low variance encountered in the Decision Tree when the depth of the tree is high, the accuracy of the Random Forest will be higher than the Decision Tree. However, for Naïve Bayes, it is easy to train and understand the results, but as its name it is based on naive assumptions that are not generally concordant with the data, which means that it requires the predictors to be independent, but in real life cases the predictors are dependent and this hinders the performance of the classifier. Besides that, Naïve Bayes classifier was also fragile to overfitting without any regularization assumption, hence Random Forest will be the most accurate classifier among them. The volume of the dataset in this experiment was too small, hence we are getting the same accuracy over three of these classifiers. With a larger dataset, the accuracy of Random Forest will be the highest compared to the other two.



## 7.0 Conclusion

In a nutshell, the objective of this project was achieved. The concentration of the ammonia in water was determined by using 2 or more algorithms. The algorithms that was used in this project are Random Forest, Decision Tree and Naïve Bayes. Three of these are classifiers, which means they will determine the concentration of the ammonia in water by classifying the data according to the concentration that was defined. There are total of two days of images for the ammonia test tubes. Those images were processed by using MATLAB to get a cleaner image for the classifier. The region of interest in the images were cropped and added with median and average filter to remove the label of the test tube and the noise of the images. Next, the mean and entropy of the processed images was determined by using the built-in function in MATLAB and these data was saved into an excel file. Three types of classifier were then implemented to classify the concentration of the ammonia in the water according to the mean and entropy of the colour of the processed images. Lastly, the accuracy of the classifier was output to show that how accurate the concentration was classified according to the features of the processed images.

## 8.0 Reference

1. Aquarium Pharmaceuticals. (n.d.). *Ammonia (Nh3/Nh4+) Test Kit Instructions*. [online] Available at: [https://www.apifishcare.com/pdf/Ammonia\\_Test\\_Kit\\_LR8600\\_Insert.pdf](https://www.apifishcare.com/pdf/Ammonia_Test_Kit_LR8600_Insert.pdf) [Accessed 13 July 2019].
2. Aquarium Pharmaceuticals. (2014). *Liquid Ammonia Test Solution #1*. [online] Available at: [https://www.apifishcare.com/pdf/API\\_LIQUID\\_AMMONIA\\_TEST\\_SOLUTION\\_1-2.pdf](https://www.apifishcare.com/pdf/API_LIQUID_AMMONIA_TEST_SOLUTION_1-2.pdf) [Accessed 13 July 2019].
3. Aquarium Pharmaceuticals. (2018). *Liquid Ammonia Test Solution #2*. [online] Available at: [https://www.apifishcare.com/pdf/API\\_LIQUID\\_AMMONIA\\_TEST\\_SOLUTION\\_2-2.pdf](https://www.apifishcare.com/pdf/API_LIQUID_AMMONIA_TEST_SOLUTION_2-2.pdf) [Accessed 13 July 2019].

4. Aqua-Fish.Net. (n.d.). *Aquariums and Ammonia, Nitrates, Nitrites - Guide, Testing & Forum*. [online] Available at: <https://en.aqua-fish.net/articles/controlling-ammonia-related-substances-aquariums> [Accessed 13 July 2019].
5. S.B. Rajesh. (2018). *Decision trees-a simple way to visualize a decision*. [online] Available at: <https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb> [Accessed 13 July 2019].
6. T. Yiu. (2019). *Understanding Random Forest: How the Algorithm Works and Why it Is So Effective*. [online] Available at: <https://towardsdatascience.com/understanding-random-forest-58381e0602d2> [Accessed 13 July 2019].
7. Xin-Yue, B., Sheng, L., Wan-Gan, S. and Hong-Wen, G. (2018). *Using a PC camera to determine the concentration of nitrite, ammonia nitrogen, sulfide, phosphate, and copper in water*. [online] *Royal Society of Chemistry 2018*. Available at: <http://pubs.rsc.org/en/content/articlelanding/2018/ay/c8ay00312b> [Accessed 14 July 2019].
8. Navlani, A. (2018) *Naive Bayes Classification using Scikit-learn* [Online]. Available at: <https://www.datacamp.com/community/tutorials/naive-bayes-scikit-learn> [Accessed 15 July 2018]
9. MathWorks. (n.d.). *What Is Image Segmentation? 3 things you need to know*. [online] Available at <https://www.mathworks.com/discovery/image-segmentation.html> [Accessed 14 July 2019].
10. Bhargavi, K., Jyothi, S. (2014). 'A Survey on Threshold Based Segmentation Technique in Image Processing'. *International Journal Of Innovative Research & Development*, 3(12), pp. 234-236.
11. Beddad, B., Hachemi, K. (2018). 'Efficient Implementation of An Improved Median Filter on TMS320C6416 Digital Signal Processor'. *2018 International Conference on Electrical Sciences and Technologies in Maghreb (CISTEM)*.
12. Sari, Y.A., Adinugroho, S. (2017). 'Tomato ripeness clustering using 6-means algorithm based on v-channel otsu segmentation'. *2017 5th International Symposium on Computational and Business Intelligence (ISCBI)*, pp. 32-36.

13. Zawbaa1, H. M., Hazman, M., Abbass, M., Hassanien, A.E. (2014). 'Automatic fruit classification using random forest algorithm'. *2014 14th International Conference on Hybrid Intelligent Systems, HIS 2014*, pp. 164-168.
14. Xu, B.X., Ye,Y.M., Lei, N. (2012). 'An Improved Random Forest Classifier for Image Classification'. *International Conference on Information and Automation Shenyang*, pp. 795-800.
15. Agarwal, S., Bhangale, N., Dhanure, K., Gavhane, S., Chakkarwar, V.A., Nagori, M.B. (2018). 'Application of Colorimetry to determine Soil Fertility through Naive Bayes Classification Algorithm'. *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*.
16. Robert F. and Simon P., 2004. *Image Processing Learning Resources*. [online] Available at: <[https://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr\\_top.htm](https://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm)> [Accessed 2 August 2019]
17. Jacci H., 2019. *The HSV Color Model in Graphic Design*. [online] Available at: <<https://www.lifewire.com/what-is-hsv-in-design-1078068>> [Accessed 2 August 2019]
18. Pooja J., 2016. *HSV Color Model*. [online] Available at: <<https://www.slideshare.net/KittuSahani/hsv-color-model>> [Accessed 2 August 2019]
19. Rafael C., Richard E. and Steven L., 2003. *Digital Image Processing Using MATLAB*. New York: Prentice Hall.
20. Esley T., Yasel G., Osvaldo P. and Roberto R., 2015. 'Behavior Study of Entropy in a Digital Image Through An Iterative Algorithm of the Mean Shift Filtering'. *International Journal of Soft Computing, Mathematics and Control*, 4(3), pp. 3.
21. Gupta, P. (2017) *Decision Trees in Machine Learning* [Online]. Available at: <https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052> [Accessed 31 July 2018]

22. Gandhi, R. (2018) *Naive Bayes Classifier* [Online]. Available at: <https://towardsdatascience.com/naive-bayes-classifier-81d512f50a7c> [Accessed 31 July 2018]
23. Hewa, K. (2018) *K-Fold Cross Validation* [Online]. Available at: <https://medium.com/datadriveninvestor/k-fold-cross-validation-6b8518070833> [Accessed 1 August 2018]

## 9.0 Appendices

```
clc;
close all;
clear all;

%concentration of ammonia
Concentration =
[0.25;0.25;0.25;0.5;0.5;0.5;1;1;1;2;2;2;4;4;4;8;8;8;0;0.25;0.25;0.25;0.5;0.5;0.5;1;1;1;2;
2;2;4;4;4;8;8;8;0];

MR = [];
MG = [];
MB = [];
MH = [];
MS = [];
MV = [];

STDR = [];
STDG = [];
STDB = [];
STDH = [];
STDS = [];
STDV = [];

ER = [];
EG = [];
EB = [];
EH = [];
```

```

ES = [];
EV = [];

folder = ["day1", "day2"];
[a, b] = size(folder);

for F=1:b
    imDir = fullfile('c:\','Users','tanli', 'OneDrive','year 4 sem 1', 'image processing',
'assignment' , folder(F));

    saveDir = fullfile('c:\','Users','tanli', 'OneDrive','year 4 sem 1', 'image processing',
'assignment' , 'processed' , folder(F));
    mkdir(saveDir);

    imds = imageDatastore(imDir);
    N = numpartitions(imds); %number of image in folder

    for n=1:N
        I = readimage(imds, n);

        if F==1
            if n == N %for control
                B = imcrop(I, [1700 1370 150 500]); %crop ROI
                Q = B;
                %figure, imshow(B)
                imwrite(B, saveDir + "\" + Concentration(19*F)+"_Crop.png");
            end
        end
    end
end
else

```

```

        A = imcrop(I, [750 1370 150 500]); %crop ROI
        B = imcrop(I, [1700 1370 150 500]); %crop ROI
        C = imcrop(I, [2750 1370 150 500]); %crop ROI

%
        %figure, imshow(A)
        %figure, imshow(B)
        %figure, imshow(C)

        %save image

        imwrite(A, saveDir + "\" + Concentration(1+(3*(n-1))) + "_" +
1+"_Crop.png");
        imwrite(B, saveDir + "\" + Concentration(2+(3*(n-1))) + "_" +
2+"_Crop.png");
        imwrite(C, saveDir + "\" + Concentration(3+(3*(n-1))) + "_" +
3+"_Crop.png");

        Q = cat(3, A, B, C);
    end
else
    if n == N %for control
        B = imcrop(I, [1700 1370 150 500]); %crop ROI
        Q = B;
%        figure, imshow(B)
        imwrite(B, saveDir + "\" + Concentration(19*(F))+"_Crop.png");

    else
        A = imcrop(I, [520 1370 150 500]); %crop ROI
        B = imcrop(I, [1700 1370 150 500]); %crop ROI

```

```

        C = imcrop(I, [2850 1370 150 500]); %crop ROI

        %figure, imshow(A)

        %figure, imshow(B)

        %figure, imshow(C)

        imwrite(A, saveDir + "\" + Concentration(1+(3*(n-1))+19) + "_" +
1+"_Crop.png");

        imwrite(B, saveDir + "\" + Concentration(2+(3*(n-1))+19) + "_" +
2+"_Crop.png");

        imwrite(C, saveDir + "\" + Concentration(3+(3*(n-1))+19) + "_" +
3+"_Crop.png");

        Q = cat(3, A, B, C);

    end

end

%median filter remove logo
for x=0:2

    if n == N
        J_R = Q(:, :, 1);
        J_G = Q(:, :, 2);
        J_B = Q(:, :, 3);
    else
        J_R = Q(:, :, 1+(3*x));
        J_G = Q(:, :, 2+(3*x));
        J_B = Q(:, :, 3+(3*x));
    end
end

```



```

J_R_F = medfilt2(J_R, [200 200], 'symmetric');
J_G_F = medfilt2(J_G, [200 200], 'symmetric');
J_B_F = medfilt2(J_B, [200 200], 'symmetric');
J = cat(3, J_R_F, J_G_F, J_B_F);

if(n == N)
    imwrite(J, saveDir + "\" + Concentration(19*F) + \"_\"+_Median.png");
else
    imwrite(J, saveDir + "\" + Concentration(1+x+(3*(n-1))+(19*(F-1))) + \"_\" +
(x+1)+\"_Median.png\");
end

%figure, imshow(J)

% 51x51 average filter to get uniform image
ave_filter=fspecial('average',[51 51]);
J=imfilter(J,ave_filter,'replicate');

if(n == N)
    imwrite(J, saveDir + "\" + Concentration(19*F)+\"_Average.png\");
else
    imwrite(J, saveDir + "\" + Concentration(1+x+(3*(n-1))+(19*(F-1))) + \"_\" +
(x+1)+\"_Average.png\");
end

%figure, imshow(J)

```

```

%convert RGB to HSV

K = rgb2hsv(J);

K_H = K(:,:,1);
K_S = K(:,:,2);
K_V = K(:,:,3);


J_R = J(:,:,1);
J_G = J(:,:,2);
J_B = J(:,:,3);


% find the mean of each channel

MR = [MR;mean2(J_R)];
MG = [MG;mean2(J_G)];
MB = [MB;mean2(J_B)];
MH = [MH;mean2(K_H)];
MS = [MS;mean2(K_S)];
MV = [MV;mean2(K_V)];


%calculate entropy

ER=[ER;entropy(J_R)];
EG=[EG;entropy(J_G)];
EB=[EB;entropy(J_B)];
EH=[EH;entropy(K_H)];
ES=[ES;entropy(K_S)];
EV=[EV;entropy(K_V)];


if n == N

```

```

        break
    end
end
end
end
end
T = table(MR,MG,MB,MH,MS,MV,ER,EG,EB,EH,ES,EV,Concentration);
filename = 'imageDataNEWTEST.xlsx';
writetable(T,filename,'Sheet',1, 'WriteVariableNames',true);

```

Code Listing 1: MATLAB code for Feature Extraction

```

x=[1,2,3,4,5,6];
x1=[1,6];

saveDir1 = fullfile('c:\','Users','tanli', 'OneDrive','year 4 sem 1', 'image processing',
'assignment', 'processed', 'Graph');
mkdir(saveDir1);

% plot the graph with mean of Red channel as y-axis and number of tube as x-axis
figure(1)
plot([MR(1:3);MR(20:22)],x,[MR(4:6);MR(23:25)],x,[MR(7:9);MR(26:28)],x,[MR(10:12);MR(29:31)],x,[MR(13:15);MR(32:34)],x,[MR(16:18);MR(35:37)],x,[MR(19);MR(38)],x1,'--')
set(gca,'ytick',0:1:6)

set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1, 0.96]); %set figure to full screen

% labelling the title and legend of the graph

```

```

title('Mean of Red channel for all concentration')

legend('concentration 0.25','concentration 0.5','concentration 1.0','concentration
2.0','concentration 4.0','concentration 8.0','concentration blank')

ylabel('Number of tube')

xlabel('Mean of Red channel')

saveas(gcf,saveDir1 +"\\" + "mean of Red channel.png");

% plot the graph with mean of Green channel as y-axis and number of tube as x-axis
figure(2)

plot([MG(1:3);MG(20:22)],x,[MG(4:6);MG(23:25)],x,[MG(7:9);MG(26:28)],x,[MG(1
0:12);MG(29:31)],x,[MG(13:15);MG(32:34)],x,[MG(16:18);MG(35:37)],x,[MG(19);
MG(38)],x1,'--')

set(gca,'ytick',0:1:6)

set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1, 0.96]); %set figure to full
screen

% labelling the title and legend of the graph

title('Mean of Green channel for all concentration')

legend('concentration 0.25','concentration 0.5','concentration 1.0','concentration
2.0','concentration 4.0','concentration 8.0','concentration blank')

ylabel('Number of tube')

xlabel('Mean of Green channel')

saveas(gcf,saveDir1 +"\\" + "mean of Green channel.png")

% plot the graph with mean of Blue channel as y-axis and number of tube as x-axis
figure(3)

plot([MB(1:3);MB(20:22)],x,[MB(4:6);MB(23:25)],x,[MB(7:9);MB(26:28)],x,[MB(10
:12);MB(29:31)],x,[MB(13:15);MB(32:34)],x,[MB(16:18);MB(35:37)],x,[MB(19);MB
(38)],x1,'--')

set(gca,'ytick',0:1:6)

```

```

set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1, 0.96]); %set figure to full
screen

% labelling the title and legend of the graph
title('Mean of Blue channel for all concentration')

legend('concentration 0.25','concentration 0.5','concentration 1.0','concentration
2.0','concentration 4.0','concentration 8.0','concentration blank')

ylabel('Number of tube')

xlabel('Mean of Blue channel')

saveas(gcf,saveDir1 +"\\" + "mean of Blue channel.png")

% plot the graph with mean of Hue as y-axis and number of tube as x-axis
figure(4)
plot([MH(1:3);MH(20:22)],x,[MH(4:6);MH(23:25)],x,[MH(7:9);MH(26:28)],x,[MH(1
0:12);MH(29:31)],x,[MH(13:15);MH(32:34)],x,[MH(16:18);MH(35:37)],x,[MH(19);
MH(38)],x1,'--')

set(gca,'ytick',0:1:6)

set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1, 0.96]); %set figure to full
screen

% labelling the title and legend of the graph
title('Mean of Hue for all concentration')

legend('concentration 0.25','concentration 0.5','concentration 1.0','concentration
2.0','concentration 4.0','concentration 8.0','concentration blank')

ylabel('Number of tube')

xlabel('Mean of Hue')

saveas(gcf,saveDir1 +"\\" + "mean of Hue.png")

% plot the graph with mean of Saturation as y-axis and number of tube as x-axis

```

```

figure(5)

plot([MS(1:3);MS(20:22)],x,[MS(4:6);MS(23:25)],x,[MS(7:9);MS(26:28)],x,[MS(10:12);MS(29:31)],x,[MS(13:15);MS(32:34)],x,[MS(16:18);MS(35:37)],x,[MS(19);MS(38)],x1,'--')

set(gca,'ytick',0:1:6)

set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1, 0.96]); %set figure to full screen


% labelling the title and legend of the graph
title('Mean of Saturation for all concentration')

legend('concentration 0.25','concentration 0.5','concentration 1.0','concentration 2.0','concentration 4.0','concentration 8.0','concentration blank')

ylabel('Number of tube')

xlabel('Mean of Saturation')

saveas(gcf,saveDir1 + "\" + "mean of Saturation.png")


% plot the graph with mean of Value as y-axis and number of tube as x-axis
figure(6)

plot([MV(1:3);MV(20:22)],x,[MV(4:6);MV(23:25)],x,[MV(7:9);MV(26:28)],x,[MV(10:12);MV(29:31)],x,[MV(13:15);MV(32:34)],x,[MV(16:18);MV(35:37)],x,[MV(19);MV(38)],x1,'--')

set(gca,'ytick',0:1:6)

set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1, 0.96]); %set figure to full screen


% labelling the title and legend of the graph
title('Mean of Saturation for all concentration')

legend('concentration 0.25','concentration 0.5','concentration 1.0','concentration 2.0','concentration 4.0','concentration 8.0','concentration blank')

ylabel('Number of tube')

```

```

xlabel('Mean of Value')

saveas(gcf,saveDir1 +"\\" + "mean of Value.png")


% plot the graph with Entropy of Red channel as y-axis and number of tube as x-axis
figure(7)

plot([ER(1:3);ER(20:22)],x,[ER(4:6);ER(23:25)],x,[ER(7:9);ER(26:28)],x,[ER(10:12);
ER(29:31)],x,[ER(13:15);ER(32:34)],x,[ER(16:18);ER(35:37)],x,[ER(19);ER(38)],x1,'
--')

set(gca,'ytick',0:1:6)

set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1, 0.96]); %set figure to full
screen


% labelling the title and legend of the graph
title('Entropy of Red channel for all concentration')

legend('concentration 0.25','concentration 0.5','concentration 1.0','concentration
2.0','concentration 4.0','concentration 8.0','concentration blank')

ylabel('Number of tube')

xlabel('Entropy of Red channel')

saveas(gcf,saveDir1 +"\\" + "Entropy of Red channel.png")


% plot the graph with Entropy of Green channel as y-axis and number of tube as x-axis
figure(8)

plot([EG(1:3);EG(20:22)],x,[EG(4:6);EG(23:25)],x,[EG(7:9);EG(26:28)],x,[EG(10:12)
;EG(29:31)],x,[EG(13:15);EG(32:34)],x,[EG(16:18);EG(35:37)],x,[EG(19);EG(38)],x
1,'--')

set(gca,'ytick',0:1:6)

set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1, 0.96]); %set figure to full
screen


% labelling the title and legend of the graph

```

```

title('Entropy of Green channel for all concentration')

legend('concentration 0.25','concentration 0.5','concentration 1.0','concentration
2.0','concentration 4.0','concentration 8.0','concentration blank')

ylabel('Number of tube')

xlabel('Entropy of Green channel')

saveas(gcf,saveDir1 +"\\" + "Entropy of Green channel.png")


% plot the graph with Entropy of Blue channel as y-axis and number of tube as x-axis
figure(9)

plot([EB(1:3);EB(20:22)],x,[EB(4:6);EB(23:25)],x,[EB(7:9);EB(26:28)],x,[EB(10:12);
EB(29:31)],x,[EB(13:15);EB(32:34)],x,[EB(16:18);EB(35:37)],x,[EB(19);EB(38)],x1,'
--')

set(gca,'ytick',0:1:6)

set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1, 0.96]); %set figure to full
screen


% labelling the title and legend of the graph
title('Entropy of Blue channel for all concentration')

legend('concentration 0.25','concentration 0.5','concentration 1.0','concentration
2.0','concentration 4.0','concentration 8.0','concentration blank')

ylabel('Number of tube')

xlabel('Entropy of Blue channel')

saveas(gcf,saveDir1 +"\\" + "Entropy of Blue channel.png")


% plot the graph with Entropy of Hue as y-axis and number of tube as x-axis
figure(10)

plot([EH(1:3);EH(20:22)],x,[EH(4:6);EH(23:25)],x,[EH(7:9);EH(26:28)],x,[EH(10:12)
;EH(29:31)],x,[EH(13:15);EH(32:34)],x,[EH(16:18);EH(35:37)],x,[EH(19);EH(38)],x
1,'--')

set(gca,'ytick',0:1:6)

```



```

set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1, 0.96]); %set figure to full
screen

% labelling the title and legend of the graph
title('Entropy of Hue for all concentration')
legend('concentration 0.25','concentration 0.5','concentration 1.0','concentration
2.0','concentration 4.0','concentration 8.0','concentration blank')

ylabel('Number of tube')
xlabel('Mean of Hue')
saveas(gcf,saveDir1 +"\\" + "Entropy of Hue.png")

% plot the graph with Entropy of Saturation as y-axis and number of tube as x-axis
figure(11)
plot([ES(1:3);ES(20:22)],x,[ES(4:6);ES(23:25)],x,[ES(7:9);ES(26:28)],x,[ES(10:12);E
S(29:31)],x,[ES(13:15);ES(32:34)],x,[ES(16:18);ES(35:37)],x,[ES(19);ES(38)],x1,'--')
set(gca,'ytick',0:1:6)

set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1, 0.96]); %set figure to full
screen

% labelling the title and legend of the graph
title('Entropy of Saturation for all concentration')
legend('concentration 0.25','concentration 0.5','concentration 1.0','concentration
2.0','concentration 4.0','concentration 8.0','concentration blank')
ylabel('Number of tube')
xlabel('Entropy of Saturation')
saveas(gcf,saveDir1 +"\\" + "Entropy of Saturation.png")

% plot the graph with Entropy of Value as y-axis and number of tube as x-axis
figure(12)

```

```

plot([EV(1:3);EV(20:22)],x,[EV(4:6);MV(23:25)],x,[EV(7:9);MV(26:28)],x,[EV(10:1
2);EV(29:31)],x,[EV(13:15);EV(32:34)],x,[EV(16:18);EV(35:37)],x,[EV(19);EV(38)],
x1,'--')

set(gca,'ytick',0:1:6)

set(gcf, 'Units', 'Normalized', 'OuterPosition', [0, 0.04, 1, 0.96]); %set figure to full
screen

% labelling the title and legend of the graph

title('Entropy of Saturation for all concentration')

legend('concentration 0.25','concentration 0.5','concentration 1.0','concentration
2.0','concentration 4.0','concentration 8.0','concentration blank')

ylabel('Number of tube')

xlabel('Entropy of Value')

saveas(gcf,saveDir1 + "\" + "Entropy of Value.png")

```

Code Listing 2: MATLAB code for Graph Plotting

```

# Random Forest Classification

# Importing the libraries

import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import statistics as st

#dataset = pd.read_excel('imageDatanew.xlsx', index_col=None, header=0)
dataset = pd.read_excel('imageDatanewTEST.xlsx', index_col=None, header=0)

```

```
#X = dataset.iloc[:,[0,1,2,3,4,5,7]].values #both day
#X = dataset.iloc[:19,[0,2,3,4]].values #day 1 only
X = dataset.iloc[19:,[0,2,3,4,8]].values #day 2 only
y = dataset.iloc[:, 12].values

#due to y are floating value, need to encode the y value
#0 = control, 1 = 0.25, 2= 0.5, 3= 1.0, 4=2.0, 5=4.0, 6=8.0
from sklearn import preprocessing
from sklearn import utils
lab_enc = preprocessing.LabelEncoder()
y = lab_enc.fit_transform(y)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

# Fitting Random Forest Classification to the Training set
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators= 500, criterion = 'entropy',
random_state= 1)

from sklearn.naive_bayes import GaussianNB
classifierNB = GaussianNB()

# Fitting Decision Tree Classification to the Training set
from sklearn.tree import DecisionTreeClassifier
classifierDT = DecisionTreeClassifier(criterion = 'entropy', random_state = 1)
```

```
from sklearn.metrics import confusion_matrix

#Initialize K-Fold of 19-Fold
from sklearn.model_selection import KFold
kf = KFold(n_splits=19, random_state=0, shuffle=True)

score=[]
scoreNB = []
scoreDT = []

cmRF = []
cmNB = []
cmDT = []

y_test_1 = []
y_test_NB1 = []
y_test_DT1 = []

y_pred_1 = []
y_pred_NB1 = []
y_pred_DT1 = []

#perform k-fold validation
from sklearn.metrics import accuracy_score
for train_index, test_index in kf.split(X):
    print("Train:", train_index, "Validation:",test_index)
```

```
X_train, X_test = X[train_index], X[test_index]
y_train, y_test = y[train_index], y[test_index]

#normalize the data (feature scaling)
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

classifier.fit(X_train, y_train)    #train the model
y_pred = classifier.predict(X_test)    #Test the model
score.append(classifier.score(X_test,y_test)) #save the accuracy
cmRF.append(confusion_matrix(y_test, y_pred)) #save the confusion matrix
y_test_1.append(y_test)    #save the prediction value
y_pred_1.append(y_pred)    #save the real value

classifierNB.fit(X_train, y_train)
y_pred_NB = classifier.predict(X_test)
scoreNB.append(classifier.score(X_test,y_test))
cmNB.append(confusion_matrix(y_test, y_pred))
y_test_NB1.append(y_test)
y_pred_NB1.append(y_pred)

classifierDT.fit(X_train, y_train)
y_pred_DT = classifier.predict(X_test)
scoreDT.append(classifier.score(X_test,y_test))
cmDT.append(confusion_matrix(y_test, y_pred))
y_test_DT1.append(y_test)
y_pred_DT1.append(y_pred)
```

```
print("Random Forest Accuracy: ", st.mean(score))  
print("Naive Bayes Accuracy: ",st.mean(scoreNB))  
print("Decision Tree Accuracy: ",st.mean(scoreDT))
```

Code Listing 3: Python Code for Training Machine Learning Models