DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING

COLLEGE OF E&ME, NUST, RAWALPINDI

EC-350 Artificial Intelligence and Decision Support System

LAB REPORT – 13

Course Instructor: Assoc Prof Dr Arsalan Shaukat

Lab Engineer: Kashaf Raheem

Student Name: **Amina Qadeer 359607**

Degree/ Syndicate: **CE 42 A**

LAB # 13

Lab Objective:

Lab Tasks:

Task1

Code:

```
threshold = 0.5

learning_rate = 0.1

dataset = [

    [1, 0, 1],

    [1, 1, 1, 1],

    [1, 1, 0, 1],

    [1, 1, 1],

]


def perceptron(training, learning_rate, threshold):

    weights = [0] * (len(training[0]) - 1)

    error = True


    while error:

        error = False


        for sample in training:

            predicted_output = sum([x * y for x, y in zip(sample[:-1], weights)])
```

```
            if predicted_output > threshold:
                output = 1
            else:
                output = 0

            if output != sample[-1]:
                error = True
                for j in range(len(sample) - 1):
                    weights[j] += learning_rate * (sample[-1] - output) * sample[j]

    return weights

def classify(test_sample, weights, threshold):
    return sum([x * y for x, y in zip(test_sample, weights)]) > threshold

accuracy = 0
weights = perceptron(dataset, learning_rate, threshold)

for sample in dataset:
    predicted_class = classify(sample[:-1], weights, threshold)
```

```
    if predicted_class == sample[-1]:
        accuracy += 1

print("Model's accuracy:", accuracy / len(dataset) * 100)
```

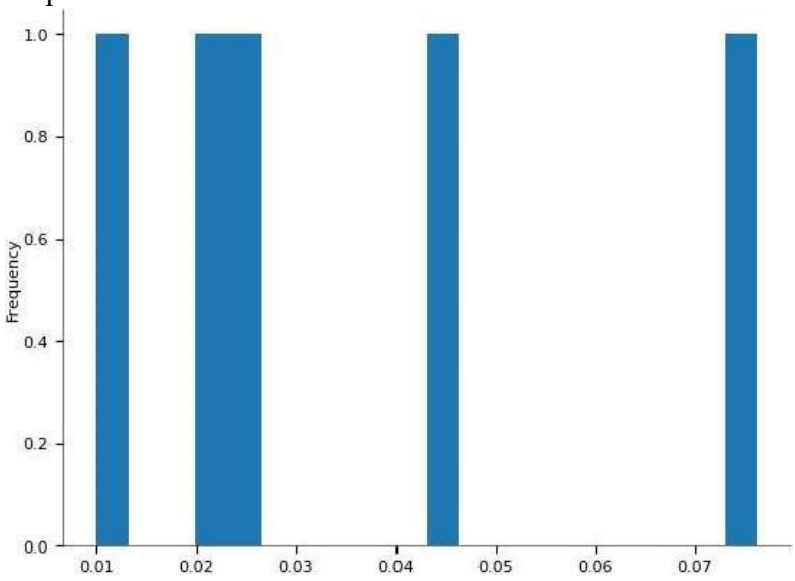
Task2:

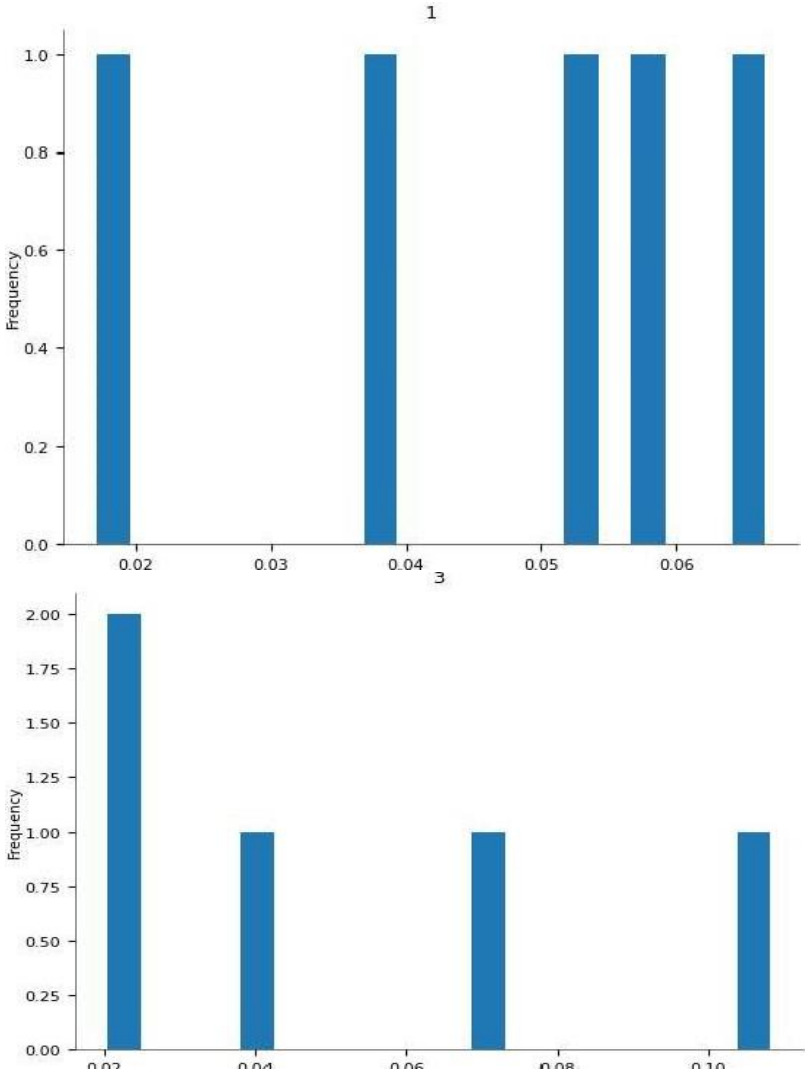
Code:

```
import pandas as pd
import math
import random
from sklearn.metrics import confusion_matrix

# Load dataset
df = pd.read_csv('/content/drive/MyDrive/Lab 12/sonar.all-data.csv', header=None)
dataset = df.values.tolist()

# Shuffle dataset
random.shuffle(dataset)

# Split dataset into training and testing sets
split_index = math.ceil(len(dataset) * 0.8)
training_dataset = dataset[:split_index]
testing_dataset = dataset[split_index:]

# Extract features and classes
features = [x[:-1] for x in training_dataset]
classes = [x[-1] for x in training_dataset]

# Set up perceptron parameters
threshold = 0.5
learning_rate = 0.1

# Train perceptron
weights = perceptron(features, classes, learning_rate, threshold)

# Initialize confusion matrix
```

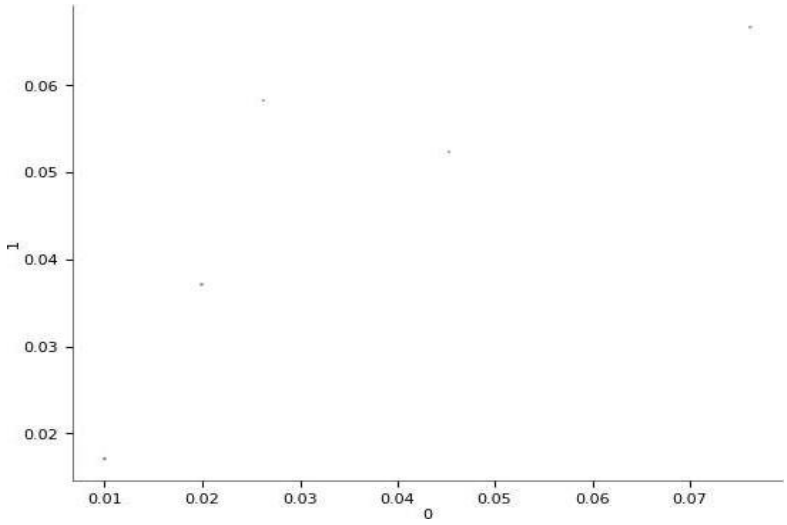
```
conf_matrix = [0, 0, 0, 0]  # [true positive, false positive, true negative, false negative]

# Evaluate perceptron on testing dataset
for sample in testing_dataset:
    predicted_class = classify(sample[:-1], weights, threshold)
    true_class = sample[-1]

    if predicted_class == true_class:
        if true_class == 1:
            conf_matrix[0] += 1  # True positive
        else:
            conf_matrix[2] += 1  # True negative
    else:
        if true_class == 1:
            conf_matrix[3] += 1  # False negative
        else:
            conf_matrix[1] += 1  # False positive

# Calculate accuracy
accuracy = (conf_matrix[0] + conf_matrix[2]) / sum(conf_matrix) * 100

# Print results
print("Accuracy:", accuracy)
print("Confusion Matrix:", conf_matrix)
```

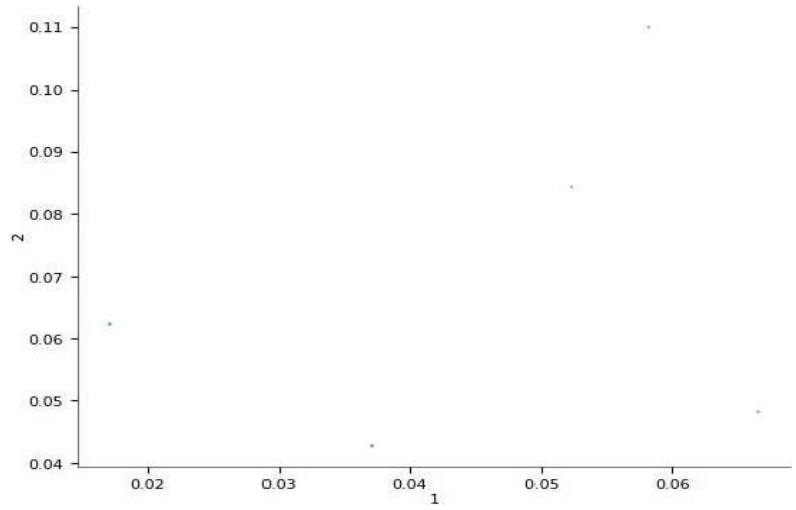
output:

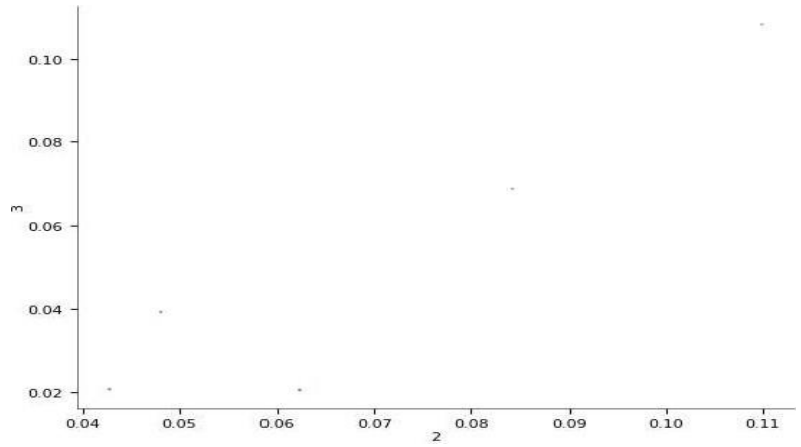


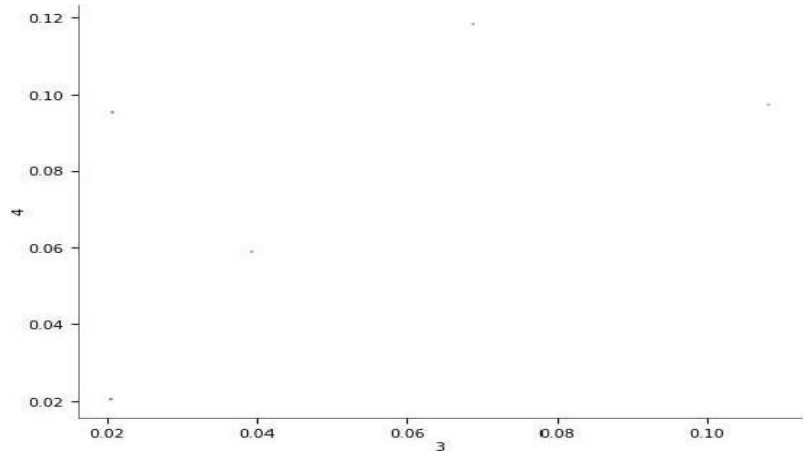


2-d distributions

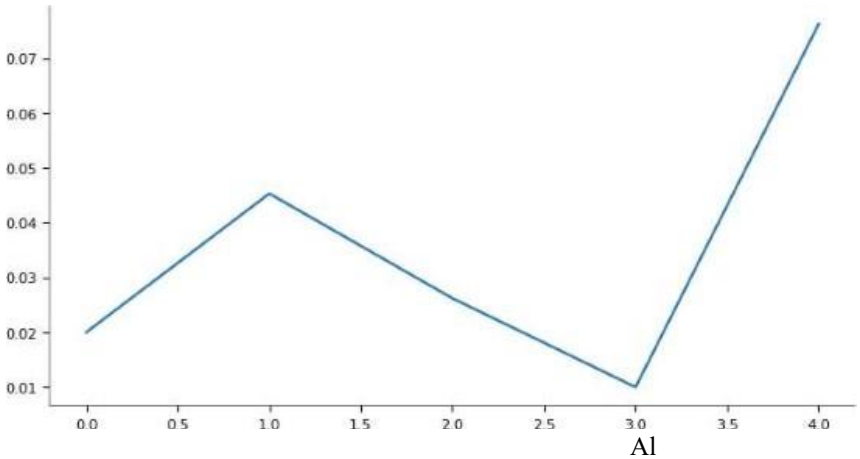



0.11
0.10
0.09
0.08
0.07
0.06
0.05
0.04
2
0.02
0.03
0.04
0.05
0.06
1

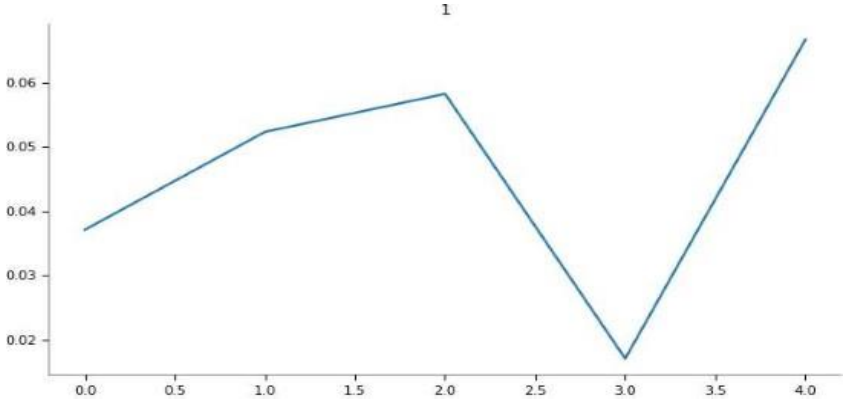

0.10
0.08
0.06
0.04
0.02
3
0.04
0.05
0.06
0.07
0.08
0.09
0.10
0.11
2

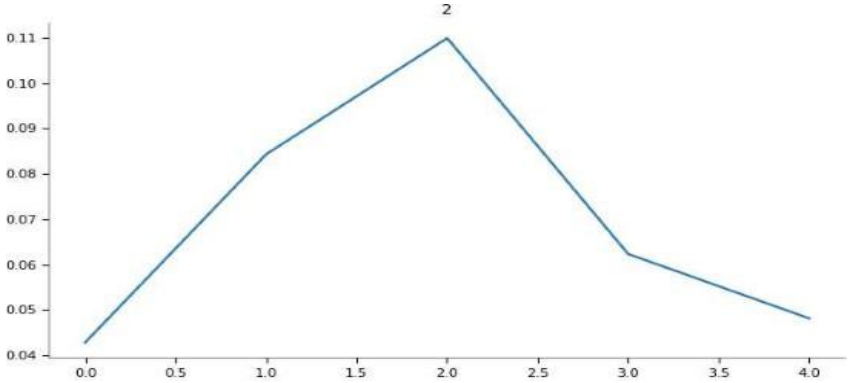

0.12
0.10
0.08
0.06
0.04
0.02
4
0.02
0.04
0.06
0.08
0.10
3

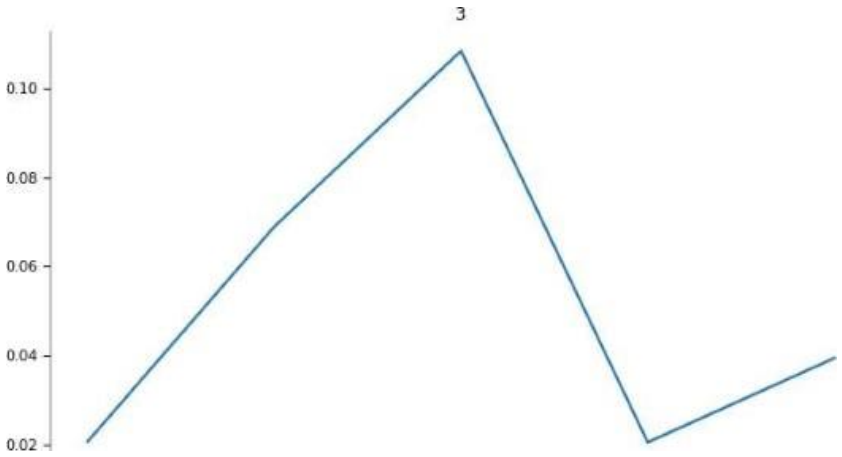
Values


0.07
0.06
0.05
0.04
0.03
0.02
0.01
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

Al


1
0.06
0.05
0.04
0.03
0.02
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


2
0.11
0.10
0.09
0.08
0.07
0.06
0.05
0.04
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


3
0.10
0.08
0.06
0.04
0.02