College of Electrical & Mechanical Engineering, NUST

NATIONAL UNIVERSITY OF SCIENCE AND TECHNOLOGY

44 CE A

AI & Decision Support System

NAME	REG NO	Syndicate
Syed Shaheer Suhaib	429292	A

Table Of Contents

Task 1

Code

```
import numpy as np
lr = 0.1
epochs = 1
W = np.zeros(3)
bias = 0
threshold =0.5

data = np.array([
    [1,0,0,1],
    [1,0,1,1],
    [1,1,1,1],
    [0,1,1,0]
]
)

train = data[:,:3]
test = data[:,-1]
features = 3
```

```
for ep_number in range(epochs+1):
    #
    print("epoch no"+str(ep_number))
    error_count = 0
    for sample in data:
        # forward pass
        output = np.dot(sample[0:3],W) + bias          # fixed slice 0:3
        if output>=threshold:
            output = 1
        else:
            output =0
        # error
        erroor =sample[3] - output
        if erroor !=0:                                 # fixed condition
            # updating
            for w_index in range(3):
                W[w_index] = W[w_index] + lr*erroor*sample[w_index]
            bias = bias+ lr*erroor
            error_count+=1

    if error_count == 0 :                               # moved outside inner
loop
        print("all corrected classified")
        break

print("weights are")
print(W)
print("bias is ")
print(bias)
```

OUTPUT:

```
PS D:\DE-44-CE-2022\mySemData\sem7\AI\lab> & "D:/soft
sem7/AI/lab/lab10/task1.py
epoch no0
epoch no1
weights are
[0.3 0.  0.1]
bias is
0.30000000000000004
PS D:\DE-44-CE-2022\mySemData\sem7\AI\lab>
```

Task 2

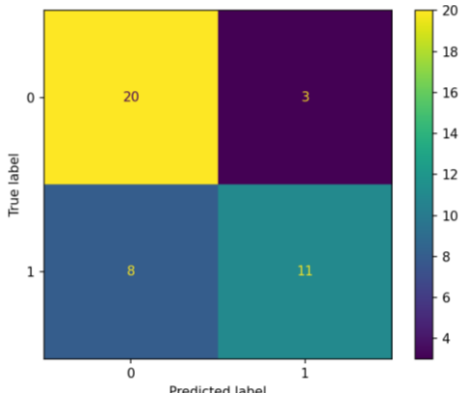
Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

lr = 0.2
epochs = 10
threshold = 0.5


df = pd.read_csv("lab10/sonar.csv")

df.iloc[:, -1] = df.iloc[:, -1].map({'R': 1, 'M': 0})

data = df.values

np.random.shuffle(data)


X = data[:, :-1]
y = data[:, -1]


split_index = int(0.8 * len(data))
X_train, X_test = X[:split_index], X[split_index:]
y_train, y_test = y[:split_index], y[split_index:]


lr = 0.1
threshold = 0.5
bias = 0
features = X_train.shape[1]
W = np.zeros(features)


while True:
```

```
    correct = 0
    for i in range(len(X_train)):
        output = np.dot(X_train[i], W) + bias
        output = 1 if output >= threshold else 0
        error = y_train[i] - output

        if error == 0:
            correct += 1
        else:
            for j in range(features):
                W[j] = W[j] + lr * error * X_train[i, j]
            bias = bias + lr * error

    train_accuracy = correct / len(X_train) * 100
    print("train accuracy:", train_accuracy)

    if train_accuracy >= 75:
        break


correct_test = 0
predictions = np.zeros(len(X_test), dtype=int)

for i in range(len(X_test)):
    output = np.dot(X_test[i], W) + bias
    output = 1 if output >= threshold else 0
    predictions[i] = output
    if output == y_test[i]:
        correct_test += 1

test_accuracy = correct_test / len(X_test) * 100
print("test accuracy:", test_accuracy)

cm = confusion_matrix(y_test.astype(int), predictions)
disp = ConfusionMatrixDisplay(confusion_matrix=cm)
disp.plot()
plt.show()
```

OUTPUT:

```
KeyboardInterrupt
PS D:\DE-44-CE-2022\mySemData\sem7\AI\lab> & "D:/software downloads
sem7/AI/lab/lab10/task2.py
train accuracy: 55.757575757575765
train accuracy: 57.57575757575758
train accuracy: 69.6969696969697
train accuracy: 73.93939393939394
train accuracy: 75.15151515151514
test accuracy: 73.80952380952381
PS D:\DE-44-CE-2022\mySemData\sem7\AI\lab>
```



Task 3

Code

```
import numpy as np
import pandas as pd


df = pd.read_csv("lab10/Iris.csv")


df["Species"] = df["Species"].map({"Iris-setosa": 1,
                                   "Iris-versicolor": 0,
                                   "Iris-virginica": 0})

data = df.values


np.random.shuffle(data)
```

```
X = data[:, 1:-1]
y = data[:, -1]


split_index = int(0.8 * len(data))
X_train = X[:split_index]
y_train = y[:split_index]
X_test = X[split_index:]
y_test = y[split_index:]


lr = 0.1
epochs = 10000
threshold = 0.5

input_size = X_train.shape[1]
hidden_size = 4
output_size = 1


W1 = np.zeros((input_size, hidden_size))
b1 = np.zeros(hidden_size)
W2 = np.zeros((hidden_size, output_size))
b2 = np.zeros(output_size)


def sigmoid(x):
    return 1 / (1 + np.exp(-x))

def d_sigmoid(x):
    return x * (1 - x)

for ep in range(epochs):

    correct = 0

    for i in range(len(X_train)):


        Z1 = np.dot(X_train[i], W1) + b1
        A1 = sigmoid(Z1)
        Z2 = np.dot(A1, W2) + b2
```

```
        A2 = sigmoid(Z2)

        # prediction check
        pred = 1 if A2 >= threshold else 0
        if pred == y_train[i]:
            correct += 1

        # error
        error = A2 - y_train[i]

        # backprop
        delta2 = error * d_sigmoid(A2)
        delta1 = np.dot(W2, delta2) * d_sigmoid(A1)

        # gradients
        grad_W2 = np.outer(A1, delta2)
        grad_b2 = delta2
        grad_W1 = np.outer(X_train[i], delta1)
        grad_b1 = delta1



        W2 = W2 - lr * grad_W2
        b2 = b2 - lr * grad_b2
        W1 = W1 - lr * grad_W1
        b1 = b1 - lr * grad_b1

    if ep % 1000 == 0:
        acc = (correct / len(X_train)) * 100
        print("epoch:", ep, "training accuracy:", acc)


train_accuracy = (correct / len(X_train)) * 100
print("final training accuracy:", train_accuracy)


correct_test = 0

for i in range(len(X_test)):
    Z1 = np.dot(X_test[i], W1) + b1
    A1 = sigmoid(Z1)
    Z2 = np.dot(A1, W2) + b2
    A2 = sigmoid(Z2)
```

```
    pred = 1 if A2 >= threshold else 0
    if pred == y_test[i]:
        correct_test += 1

test_accuracy = (correct_test / len(X_test)) * 100
print("testing accuracy:", test_accuracy)
```

OUTPUT:

```
42
43
PROBLEMS  OUTPUT  TERMINAL  PORTS  GITLENS
PS D:\DE-44-CE-2022\mySemData\sem7\AI\lab> & "D:/software down
sem7/AI/lab/lab10/task3.py
epoch: 0 training accuracy: 64.16666666666667
epoch: 1000 training accuracy: 100.0
epoch: 2000 training accuracy: 100.0
epoch: 3000 training accuracy: 100.0
```