In [ ]:
```
Question .1
```

# Name :- Prekshita Vasudeo Patil
# Registration Number :- 20MAI0073
# Git Hub Link :- https://github.com/prekshita19/DL-Assignments

In [1]:
```
inputs = [[2,3,4],
        [8,2,5],
        [2,6,2],
        [4,2,3]]
targets=[0,0,1,0]
```

In [2]:
```
print(inputs)
```

```
[[2, 3, 4], [8, 2, 5], [2, 6, 2], [4, 2, 3]]
```

In [3]:
```
print(targets)
```

```
[0, 0, 1, 0]
```

In [4]:
```
epochs = 5
learning_rate = 0.02
weight_1 = 0
weight_2 = 0
weight_3 =  0
unit_bias = 1
bias = 0
ypredict = 0
```

In [5]:
```
for epc in range(0,epochs):
    print("\n\n\nEpoch-"+str(epc+1))
    print("-------------------------------------------------------------------")
    for i,x in zip(inputs,targets):
        input_1 = i[0]
        input_2 = i[1]
        input_3 = i[2]
        y_in = bias + (input_1 * weight_1)+(input_2*weight_2)+(input_3*weight_3)
        if y_in > 0:
            y=1
        elif y_in == 0:
            y=0
        elif y_in <0 :
            y=-1
        else:
            print("something went wrong")
        print("input_1 : - ",input_1)
        print("input_2 :- ",input_2)
        print("input_3 :-",input_3)
        print("Unit_bias :-",1)
        print("Target :- ",x)
        if y!=x:
            weight_1 =( learning_rate * x * input_1) + weight_1
            weight_2 =( learning_rate * x * input_2) + weight_2
            weight_3 =( learning_rate * x * input_3) + weight_3
            weight_1_old,weight_2_old,weight_3_old,bias_old = weight_1,weight_2,w
            bias = bias + (learning_rate * x)
            print("Y_in :-",y_in)
            print("Y :-",y)
            print("Target :-",x)
            print("Change in Weight-1 :- ",str(learning_rate) +" X  "+str(x) +' )
            print("Change in Weight-2 :- ",str(learning_rate) +" X  "+str(x) +' )
            print("Change in Weight-3 :- ",str(learning_rate) +" X  "+str(x) +' )
            print("Change in bias :- ",learning_rate * x)
            print("New Weight_1 :- ",str(learning_rate) +" X  "+str(x) +' X ' +st
            print("New Weight_1 :- ",str(learning_rate) +" X  "+str(x) +' X ' +st
            print("New Weight_1 :- ",str(learning_rate) +" X  "+str(x) +' X ' +st
            print("New Bias ",str(bias_old) +" + "+ str(learning_rate * x))
        else:
            weight_1 = weight_1
            weight_2 = weight_2
            weight_3 = weight_3
            ypredict = ypredict + 1
            print("New Weight_1 :- ",'old weight_1 = ',weight_1)
            print("New Weight_2 :- ",'old weight_2 = ',weight_2)
            print("New Weight_3 :- ",'old weight_3 = ',weight_3)
            print("New Bias :- ",'bias =',bias)
        print("\n\n")
```

```
Epoch-1
-------------------------------------------------------------------
input_1 : -  2
input_2 :-  3
input_3 :- 4
```

```
Unit_bias :- 1
Target :-  0
New Weight_1 :-  old weight_1 =  0
New Weight_2 :-  old weight_2 =  0
New Weight_3 :-  old weight_3 =  0
New Bias :-  bias = 0



input_1 : -  8
input_2 :-  2
input_3 :- 5
Unit_bias :- 1
```

In [6]:
```
print("Accuracy Score :- ",ypredict,"/",(len(targets))*epochs,"=",(ypredict/(len(
print((ypredict/(len(targets)*epochs))*100,"%")
```

```
Accuracy Score :-  6 / 20 = 0.3
30.0 %
```

In [ ]:

In [ ]:

# Name :- Prekshita vasudeo patil

# registration No:- 20MAI0073

# Github link :- https://github.com/prekshita19/DL-Assignments (https://github.com/prekshita19/DL-Assignments)

In [1]:
```
import pandas as pd
read = pd.read_csv('wheat-seeds.csv',names=['area',"perimeter",'compactness','ler
```

In [2]:
```
read.head()
```

Out[2]:

| | area | perimeter | compactness | length of kernel | width of kernel | asymmetry coefficient | length of kernel groove | Class |
|---|---|---|---|---|---|---|---|---|
| **0** | 15.26 | 14.84 | 0.8710 | 5.763 | 3.312 | 2.221 | 5.220 | 1 |
| **1** | 14.88 | 14.57 | 0.8811 | 5.554 | 3.333 | 1.018 | 4.956 | 1 |
| **2** | 14.29 | 14.09 | 0.9050 | 5.291 | 3.337 | 2.699 | 4.825 | 1 |
| **3** | 13.84 | 13.94 | 0.8955 | 5.324 | 3.379 | 2.259 | 4.805 | 1 |
| **4** | 16.14 | 14.99 | 0.9034 | 5.658 | 3.562 | 1.355 | 5.175 | 1 |

In [3]:
```
x=read[['area','perimeter','compactness','length of kernel','width of kernel','as
y=read[['Class']]
```

In [4]:
```
x.head()
```

Out[4]:

| | area | perimeter | compactness | length of kernel | width of kernel | asymmetry coefficient | length of kernel groove |
|---|---|---|---|---|---|---|---|
| **0** | 15.26 | 14.84 | 0.8710 | 5.763 | 3.312 | 2.221 | 5.220 |
| **1** | 14.88 | 14.57 | 0.8811 | 5.554 | 3.333 | 1.018 | 4.956 |
| **2** | 14.29 | 14.09 | 0.9050 | 5.291 | 3.337 | 2.699 | 4.825 |
| **3** | 13.84 | 13.94 | 0.8955 | 5.324 | 3.379 | 2.259 | 4.805 |
| **4** | 16.14 | 14.99 | 0.9034 | 5.658 | 3.562 | 1.355 | 5.175 |

In [5]:
```
y.head()
```

Out[5]:

| | Class |
|---|---|
| **0** | 1 |
| **1** | 1 |
| **2** | 1 |
| **3** | 1 |
| **4** | 1 |

In [6]:
```
print(x.shape)
print(y.shape)
```

```
(210, 7)
(210, 1)
```

In [7]:
```
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score,plot_confusion_matrix,accuracy_score
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.1,random_state = 0)
```

In [8]:
```
print('xtrain.shape :- ',xtrain.shape)
print('xtest.shape :- ',xtest.shape)
print('ytrain.shape :- ',ytrain.shape)
print('xtest.shape :- ',ytest.shape)
```

```
xtrain.shape :-  (189, 7)
xtest.shape :-  (21, 7)
ytrain.shape :-  (189, 1)
xtest.shape :-  (21, 1)
```

In [9]:
```
model = MLPClassifier((2,200),max_iter=100)
model.fit(xtrain,ytrain)
ypredict = model.predict(xtest)
print(ypredict)
```

```
C:\Users\LENOVO\anaconda3\lib\site-packages\sklearn\utils\validation.py:73: Dat
aConversionWarning: A column-vector y was passed when a 1d array was expected.
Please change the shape of y to (n_samples, ), for example using ravel().
  return f(**kwargs)

[2 3 1 1 3 1 1 2 1 3 1 2 3 3 1 1 3 1 1 1 3]

C:\Users\LENOVO\anaconda3\lib\site-packages\sklearn\neural_network\_multilayer_
perceptron.py:582: ConvergenceWarning: Stochastic Optimizer: Maximum iterations
(100) reached and the optimization hasn't converged yet.
  warnings.warn(
```

In [10]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
print(confusion_matrix(ypredict,ytest))
```

```
[[6 5 0]
 [1 1 1]
 [1 1 5]]
```

In [11]:
```
print(accuracy_score(ypredict,ytest))
```

```
0.5714285714285714
```

# Name :- Prekshita vasudeo patil

# registration No:- 20MAI0073

**Github link :-** https://github.com/prekshita19/DL-Assignments (https://github.com/prekshita19/DL-Assignments)

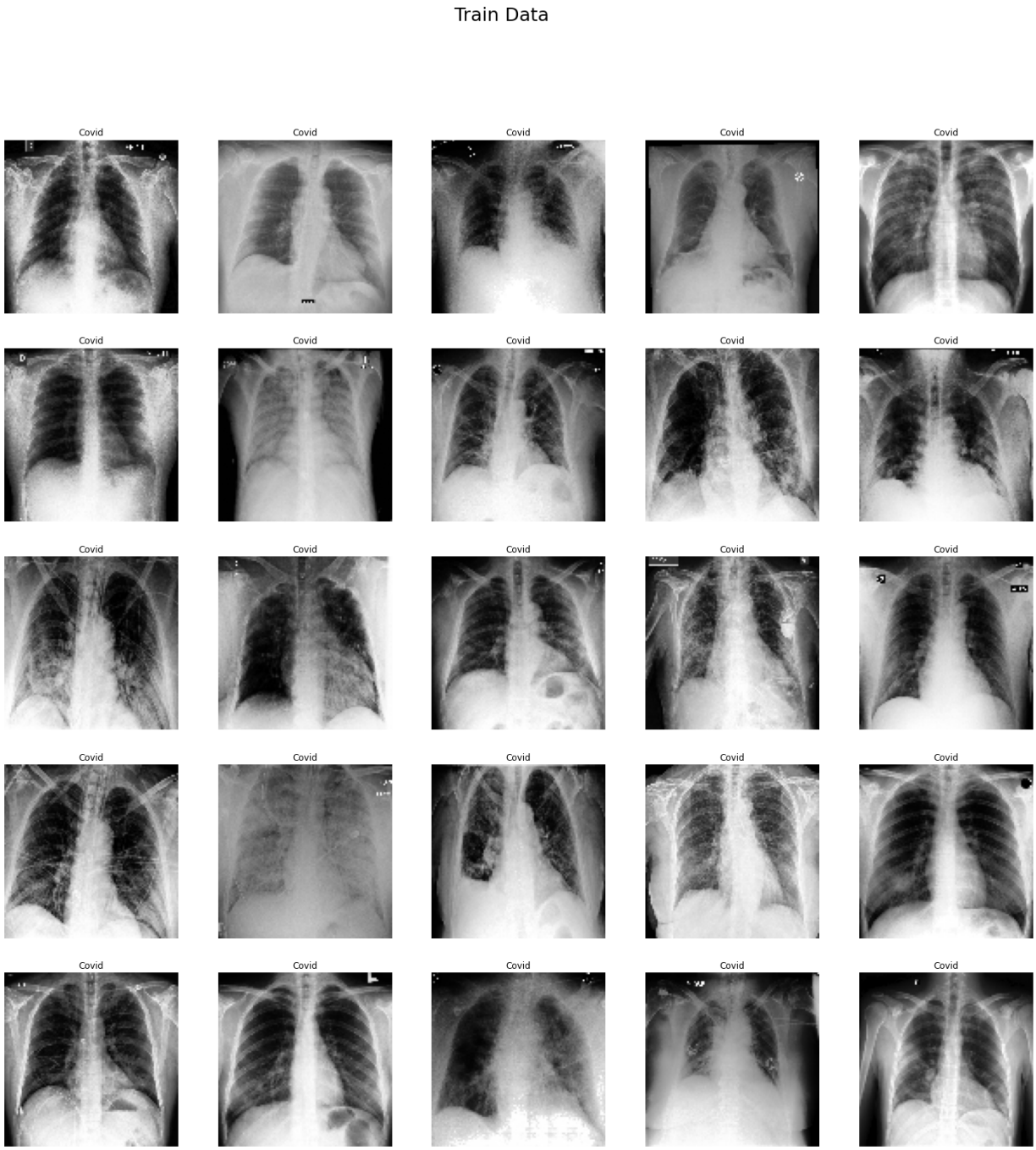
**Dataset Link :-** https://drive.google.com/drive/folders/1PTIdyrKSsLp1CRJMb_GHhJrO6MjZUbPX?usp=sharing (https://drive.google.com/drive/folders/1PTIdyrKSsLp1CRJMb_GHhJrO6MjZUbPX?usp=sharing)

In [3]:
```
import os
import cv2
import pandas as pd
from matplotlib import pyplot as plt
```

In [4]:
```
xtrain,xtest,ytrain,ytest=[],[],[],[]
for i in os.listdir('Images'):
    for j in os.listdir("Images/"+i):
        for k in os.listdir("Images/"+i+"/"+j):
            image = cv2.imread("Images/"+i+"/"+j+"/"+k)
            image = cv2.resize(image/255,(100,100))
#             image = cv2.resize(image/255,(1080,1900))
            if i=='train':
                xtrain.append(image)
                ytrain.append(j)
            elif i=='test':
                xtest.append(image)
                ytest.append(j)
            else:
                print("Something went wrong")
```

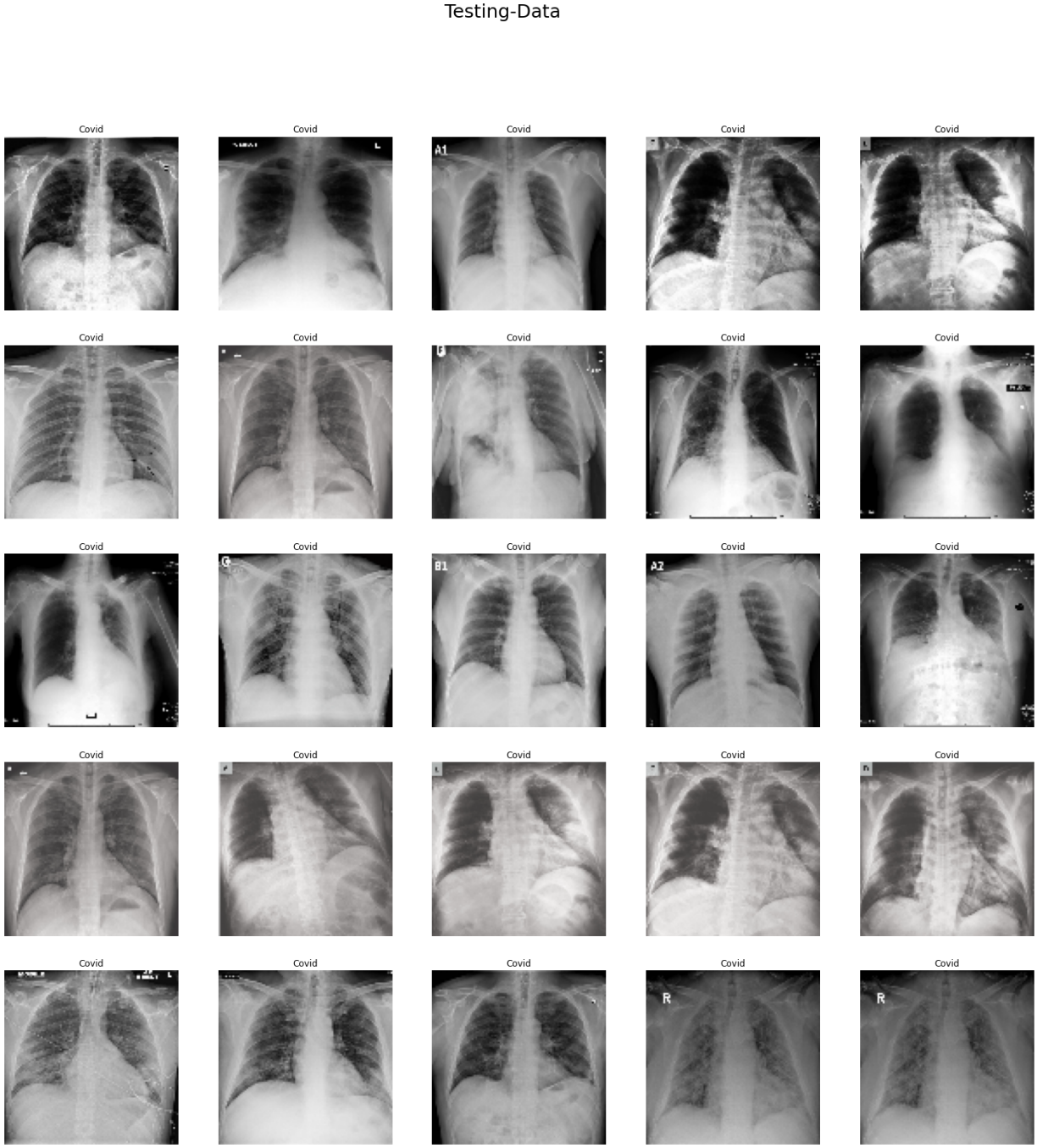
In [5]:
```
fig = plt.figure()
_, axs = plt.subplots(5, 5, figsize=(25,25))
axs = axs.flatten()
for img, ax,k in zip(xtrain, axs,ytrain):
    ax.axis("off")
    ax.set_title(k)
    ax.imshow(img)
plt.suptitle('Train Data',fontsize=25)
plt.savefig("Training Image.jpg")
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



In [6]:
```
fig = plt.figure()
_, axs = plt.subplots(5, 5, figsize=(25,25))
axs = axs.flatten()
for img, ax,k in zip(xtest, axs,ytest):
    ax.axis("off")
    ax.set_title(k)
    ax.imshow(img)
plt.suptitle('Testing-Data',fontsize=25)
plt.savefig("Testing Image.jpg")
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



In [7]:
```
import numpy as np
xtrain = np.array(xtrain).astype(np.float32)
xtest = np.array(xtest).astype(np.float32)
```

In [8]:
```
xtrain[-1].shape
```

Out[8]:
```
(100, 100, 3)
```

In [9]:
```
plt.imshow(xtrain[0])
print(xtrain[0].shape)
```

```
(100, 100, 3)
```


In [10]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
ytrain=le.fit_transform(ytrain)
ytest=le.fit_transform(ytest)
```

In [11]:
```
ytest
```

Out[11]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2],
      dtype=int64)
```

In [12]:
```
xtrain_reshaped = xtrain.reshape((len(xtrain),-1))
xtest_reshaped = xtest.reshape((len(xtest),-1))
```

In [13]:
```
from sklearn.linear_model import Perceptron
model = Perceptron()
model.fit(xtrain_reshaped,ytrain)
predict = model.predict(xtest_reshaped)
```

In [14]:
```
from sklearn.metrics import accuracy_score,classification_report,confusion_matri
print(accuracy_score(predict,ytest)*100)
```

```
78.78787878787878
```

In [15]:
```
print(classification_report(predict,ytest))
```

```
              precision    recall  f1-score   support

           0       0.96      0.86      0.91        29
           1       0.50      0.91      0.65        11
           2       0.85      0.65      0.74        26

    accuracy                           0.79        66
   macro avg       0.77      0.81      0.76        66
weighted avg       0.84      0.79      0.80        66
```

In [16]:
```
confusion_matrix(predict,ytest)
```

Out[16]:
```
array([[25,  1,  3],
       [ 1, 10,  0],
       [ 0,  9, 17]], dtype=int64)
```