COURSE OUTCOME 4

PROGRAM - 11

AIM:

Programs on feedforward network to classify any standard dataset available in the public domain.

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from tensorflow.keras.optimizers import SGD
from matplotlib import pyplot as plt
```

```
(x_train,y_train),(x_valid,y_valid) = mnist.load_data()
```

```
x_train.shape
```

```
(60000, 28, 28)
```

```
type(x_train)
```

```
numpy.ndarray
```

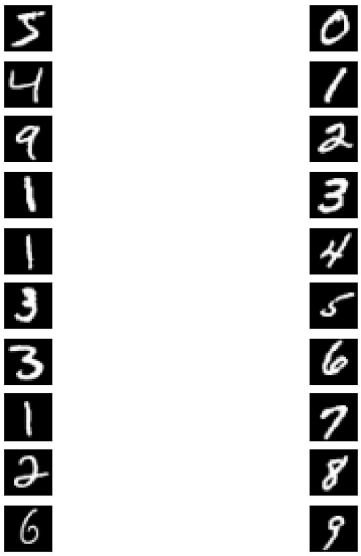
```
y_train.shape
```

```
(60000,)
```

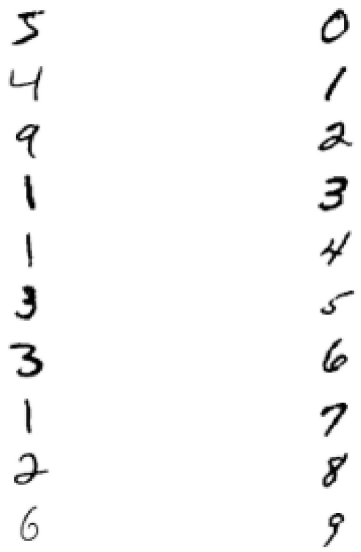
```
y_train[0:12]
```

```
array([5, 0, 4, 1, 9, 2, 1, 3, 1, 4, 3, 5], dtype=uint8)
```

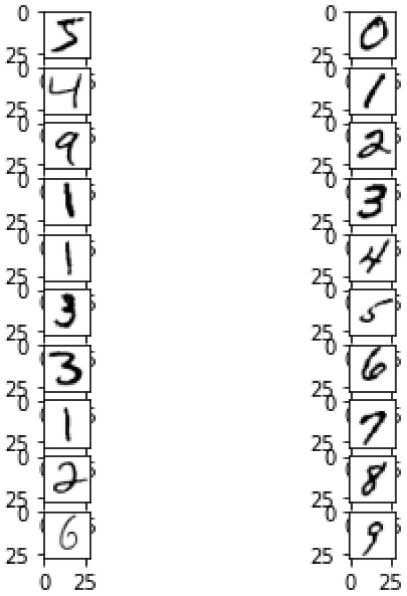
```
plt.figure(figsize=(5,5))
for k in range(20):
  plt.subplot(10,2,k+1)
  plt.imshow(x_train[k],cmap='Greys_r')
  plt.axis('off')
plt.show()
```



```
plt.figure(figsize=(5,5))
for k in range(20):
  plt.subplot(10,2,k+1)
  plt.imshow(x_train[k],cmap='Greys')
  plt.axis('off')
plt.show()
```

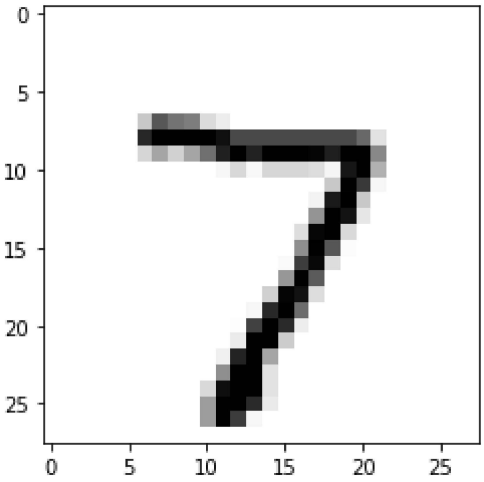


```
plt.figure(figsize=(5,5))
for k in range(20):
  plt.subplot(10,2,k+1)
  plt.imshow(x_train[k],cmap='Greys')
  plt.axis('on')
plt.show()
```



```
plt.imshow(x_valid[0],cmap='Greys')
```

```
<matplotlib.image.AxesImage at 0x7fc24c185250>
```



```
y_valid.shape
```

```
(10000,)
```

```
x_valid[0]
```

```
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,   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,   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,   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,   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,
   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,   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,   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,   0,   0,  84, 185, 159, 151,  60,  36,   0,
   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0, 222, 254, 254, 254, 254, 241, 198,
 198, 198, 198, 198, 198, 198, 198, 170,  52,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,  67, 114,  72, 114, 163, 227, 254,
 225, 254, 254, 254, 250, 229, 254, 254, 140,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  17,  66,
  14,  67,  67,  67,  59,  21, 236, 254, 106,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,   0,  83, 253, 209,  18,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0,  22, 233, 255,  83,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0, 129, 254, 238,  44,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,  59, 249, 254,  62,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0, 133, 254, 187,   5,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   9, 205, 248,  58,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0, 126, 254, 182,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,  75, 251, 240,  57,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

```
y_valid[0]
```

```
7
```

PREPROCESS DATA

```
x_train = x_train.reshape(60000,784).astype('float32')
x_valid = x_valid.reshape(10000,784).astype('float32')
```

```
x_train /= 255
x_valid /= 255
```

```
x_valid[0]
```

```
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.        ,
       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.        ,
       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.        ,
       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.        ,
       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.        ,
       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.        ,
       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.        ,
           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.        ,
           0.        , 0.        , 0.32941177, 0.7254902 , 0.62352943,
           0.5921569 , 0.23529412, 0.14117648, 0.        , 0.        ,
           0.        , 0.        , 0.        , 0.        , 0.        ,
           0.        , 0.        , 0.        , 0.        , 0.        ,
           0.        , 0.        , 0.        , 0.        , 0.        ,
           0.        , 0.        , 0.        , 0.        , 0.        ,
           0.87058824, 0.99607843, 0.99607843, 0.99607843, 0.99607843,
           0.94509804, 0.7764706 , 0.7764706 , 0.7764706 , 0.7764706 ,
           0.7764706 , 0.7764706 , 0.7764706 , 0.7764706 , 0.6666667 ,
           0.20392157, 0.        , 0.        , 0.        , 0.        ,
           0.        , 0.        , 0.        , 0.        , 0.        ,
           0.        , 0.        , 0.        , 0.2627451 , 0.44705883,
           0.28235295, 0.44705883, 0.6392157 , 0.8901961 , 0.99607843,
           0.88235295, 0.99607843, 0.99607843, 0.99607843, 0.98039216,
           0.8980392 , 0.99607843, 0.99607843, 0.54901963, 0.        ,
           0.        , 0.        , 0.        , 0.        , 0.        ,
           0.        , 0.        , 0.        , 0.        , 0.        ,
           0.        , 0.        , 0.        , 0.        , 0.        ,
```

```
x_train[0]
```

```
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.        ,
       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.        ,
       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.        ,
       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.        ,
       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.        ,
```

```
          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.        ,
          0.        , 0.        , 0.01176471, 0.07058824, 0.07058824,
          0.07058824, 0.49411765, 0.53333336, 0.6862745 , 0.10196079,
          0.6509804 , 1.        , 0.96862745, 0.49803922, 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.11764706, 0.14117648, 0.36862746, 0.6039216 ,
          0.6666667 , 0.99215686, 0.99215686, 0.99215686, 0.99215686,
          0.99215686, 0.88235295, 0.6745098 , 0.99215686, 0.9490196 ,
          0.7647059 , 0.2509804 , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.19215687, 0.93333334,
          0.99215686, 0.99215686, 0.99215686, 0.99215686, 0.99215686,
          0.99215686, 0.99215686, 0.99215686, 0.9843137 , 0.3647059 ,
          0.32156864, 0.32156864, 0.21960784, 0.15294118, 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.07058824, 0.85882354, 0.99215686, 0.99215686,
          0.99215686, 0.99215686, 0.99215686, 0.7764706 , 0.7137255 ,
          0.96862745, 0.94509804, 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.3137255 , 0.6117647 , 0.41960785, 0.99215686, 0.99215686,
          0.8039216 , 0.04313726, 0.        , 0.16862746, 0.6039216 ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.05490196,
```

```
from keras import utils as np_utils
n_classes = 10
y_train = keras.utils.np_utils.to_categorical(y_train,n_classes)
y_valid = keras.utils.np_utils.to_categorical(y_valid,n_classes)
```

```
y_valid[0]
```

```
array([0., 0., 0., 0., 0., 0., 0., 1., 0., 0.], dtype=float32)
```

```
y_train[0]
```

```
array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.], dtype=float32)
```

```
x_valid[3]
```

```
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.        ,
       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.        ,
       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.        ,
       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.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.04313726,
       0.5882353 , 0.99215686, 0.7921569 , 0.12156863, 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.14509805, 0.9843137 , 0.9843137 ,
       0.99215686, 0.41960785, 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.08235294,
       0.77254903, 0.9843137 , 0.9843137 , 0.99215686, 0.41960785,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.43137255, 0.74509805, 0.9843137 , 0.9843137 ,
       0.9843137 , 0.99215686, 0.6627451 , 0.42745098, 0.24313726,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.99215686,
       0.9843137 , 0.9843137 , 0.9843137 , 0.9843137 , 0.99215686,
       0.9843137 , 0.9843137 , 0.8627451 , 0.2       , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
       0.        , 0.        , 0.        , 0.        , 0.        ,
```

```
        0.        , 0.7137255 , 1.        , 0.99215686, 0.99215686,
        0.99215686, 0.99215686, 0.91764706, 0.87058824, 0.99215686,
        0.99215686, 0.99215686, 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.        , 0.        ,
        0.        , 0.        , 0.        , 0.24705882, 0.8666667 ,
```

```
y_valid[3]
```

```
array([1., 0., 0., 0., 0., 0., 0., 0., 0., 0.], dtype=float32)
```

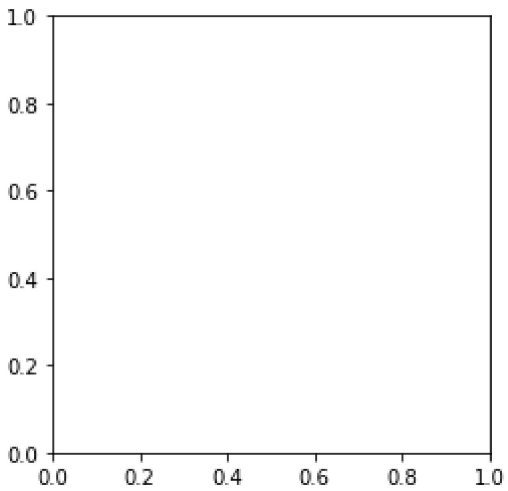
```
plt.imshow(x_valid[3],cmap='Greys')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-272-90f8c474b596> in <module>()
----> 1 plt.imshow(x_valid[3],cmap='Greys')

                                5 frames
/usr/local/lib/python3.7/dist-packages/matplotlib/image.py in set_data(self,
    697                 or self._A.ndim == 3 and self._A.shape[-1] in [3, 4]
    698             raise TypeError("Invalid shape {} for image data"
--> 699                             .format(self._A.shape))
    700
    701         if self._A.ndim == 3:

TypeError: Invalid shape (784,) for image data
```

SEARCH STACK OVERFLOW



BUILD A MODEL

```
model = Sequential()
```

```
model.add(Dense(64, activation='sigmoid', input_shape=(784,)))

model.add(Dense(10, activation='softmax'))  #final layer

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 64)                50240

 dense_1 (Dense)             (None, 10)                650

=================================================================
Total params: 50,890
Trainable params: 50,890
Non-trainable params: 0
_________________________________________________________________
```

```
(64*784) #wi xi total weights at input
```

```
50176
```

```
(64*784)+64
```

```
50240
```

```
(10*64)
```

```
640
```

```
(10*64)+10
```

```
650
```

```
(64*784)+64 + (10*64)+10
```

```
50890
```

```
model.compile(loss = 'mean_squared_error', optimizer=SGD(learning_rate=0.2),metri
```

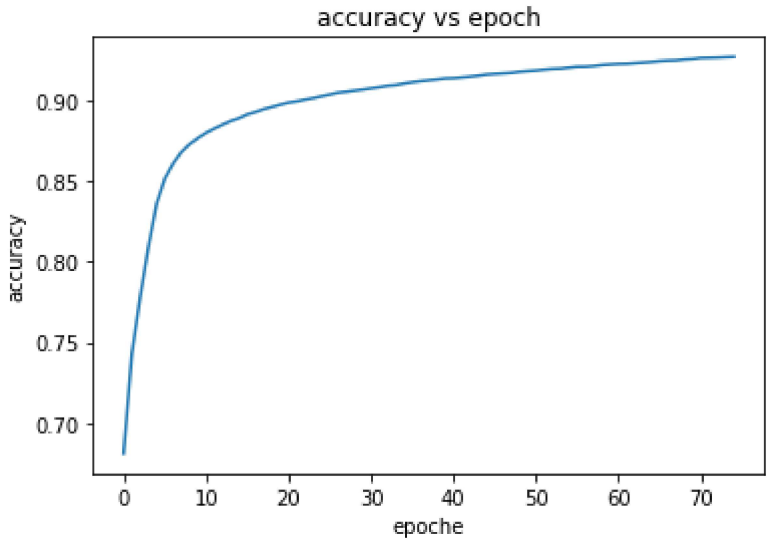
```
history=model.fit(x_train,y_train,batch_size=128,epochs=75,verbose=1)
#verbose=0 will show you nothing
#verbose = 1 will show an animated progress bar
```

```
Epoch 1/75
469/469 [==============================] - 2s 4ms/step - loss: 0.0555 - a
Epoch 2/75
469/469 [==============================] - 2s 4ms/step - loss: 0.0479 - a
Epoch 3/75
469/469 [==============================] - 2s 4ms/step - loss: 0.0417 - a
Epoch 4/75
469/469 [==============================] - 1s 3ms/step - loss: 0.0369 - a
Epoch 5/75
469/469 [==============================] - 1s 3ms/step - loss: 0.0331 - a
Epoch 6/75
469/469 [==============================] - 1s 3ms/step - loss: 0.0301 - a
Epoch 7/75
469/469 [==============================] - 2s 4ms/step - loss: 0.0277 - a
Epoch 8/75
469/469 [==============================] - 1s 3ms/step - loss: 0.0258 - a
Epoch 9/75
469/469 [==============================] - 1s 3ms/step - loss: 0.0243 - a
Epoch 10/75
469/469 [==============================] - 1s 3ms/step - loss: 0.0231 - a
Epoch 11/75
469/469 [==============================] - 1s 3ms/step - loss: 0.0221 - a
Epoch 12/75
469/469 [==============================] - 1s 2ms/step - loss: 0.0212 - a
Epoch 13/75
469/469 [==============================] - 2s 4ms/step - loss: 0.0205 - a
Epoch 14/75
469/469 [==============================] - 2s 4ms/step - loss: 0.0199 - a
Epoch 15/75
469/469 [==============================] - 1s 3ms/step - loss: 0.0193 - a
Epoch 16/75
469/469 [==============================] - 1s 3ms/step - loss: 0.0188 - a
Epoch 17/75
469/469 [==============================] - 2s 3ms/step - loss: 0.0184 - a
Epoch 18/75
469/469 [==============================] - 1s 3ms/step - loss: 0.0180 - a
Epoch 19/75
469/469 [==============================] - 1s 3ms/step - loss: 0.0177 - a
Epoch 20/75
469/469 [==============================] - 1s 3ms/step - loss: 0.0174 - a
Epoch 21/75
469/469 [==============================] - 1s 3ms/step - loss: 0.0171 - a
Epoch 22/75
469/469 [==============================] - 1s 3ms/step - loss: 0.0168 - a
Epoch 23/75
469/469 [==============================] - 1s 3ms/step - loss: 0.0166 - a
Epoch 24/75
469/469 [==============================] - 1s 2ms/step - loss: 0.0163 - a
```

```
Epoch 25/75
469/469 [==============================] - 1s 2ms/step - loss: 0.0161 - a
Epoch 26/75
469/469 [==============================] - 1s 2ms/step - loss: 0.0159 - a
Epoch 27/75
469/469 [==============================] - 1s 3ms/step - loss: 0.0157 - a
Epoch 28/75
469/469 [==============================] - 1s 2ms/step - loss: 0.0156 - a
Epoch 29/75
```

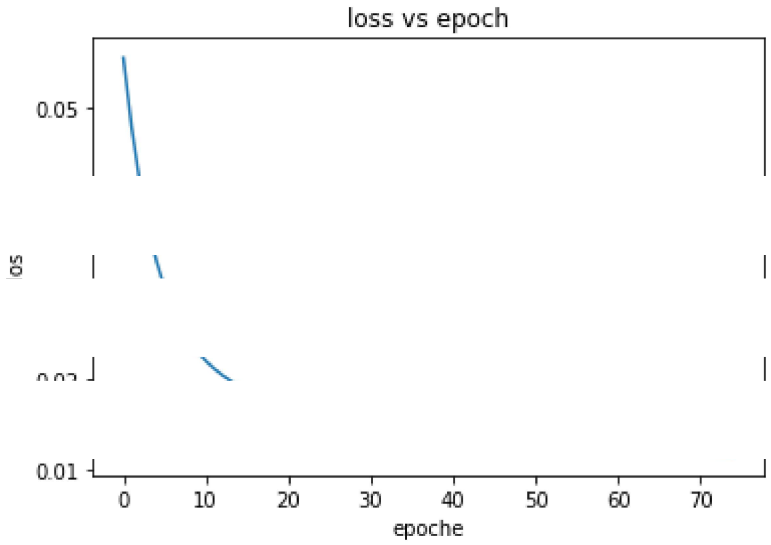
```
plt.plot(history.history['accuracy'])
plt.title('accuracy vs epoch')
plt.ylabel('accuracy')
plt.xlabel('epoche')
```

```
Text(0.5, 0, 'epoche')
```




```
plt.plot(history.history['loss'])
plt.title('loss vs epoch')
plt.ylabel('loss')
plt.xlabel('epoche')
```

```
Text(0.5, 0, 'epoche')
```



DATASET:

data.csv

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from tensorflow.keras.optimizers import SGD
from matplotlib import pyplot as plt
import numpy as np
import seaborn as sb
import pandas as pd
```

```
df = pd.read_csv('/content/data.csv')
df
```

```
           id diagnosis  ...  fractal_dimension_worst  Unnamed: 32
0      842302         M  ...                  0.11890          NaN
1      842517         M  ...                  0.08902          NaN
2    84300903         M  ...                  0.08758          NaN
3    84348301         M  ...                  0.17300          NaN
4    84358402         M  ...                  0.07678          NaN
..        ...       ...  ...                      ...          ...
564    926424         M  ...                  0.07115          NaN
565    926682         M  ...                  0.06637          NaN
566    926954         M  ...                  0.07820          NaN
567    927241         M  ...                  0.12400          NaN
568     92751         B  ...                  0.07039          NaN

[569 rows x 33 columns]
```

```
df.isnull().sum()
```

```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
```

```
smoothness_se                0
compactness_se               0
concavity_se                 0
concave points_se            0
symmetry_se                  0
fractal_dimension_se         0
radius_worst                 0
texture_worst                0
perimeter_worst              0
area_worst                   0
smoothness_worst             0
compactness_worst            0
concavity_worst              0
concave points_worst         0
symmetry_worst               0
fractal_dimension_worst      0
Unnamed: 32                569
dtype: int64
```

```
#droping feature
df.drop(['Unnamed: 32','id'],axis=1,inplace=True)
```

```
df
```

```
    diagnosis  radius_mean  ...  symmetry_worst
fractal_dimension_worst
0           M        17.99  ...          0.4601
0.11890
1           M        20.57  ...          0.2750
0.08902
2           M        19.69  ...          0.3613
0.08758
3           M        11.42  ...          0.6638
0.17300
4           M        20.29  ...          0.2364
0.07678
..        ...          ...  ...             ...                    .
..
564         M        21.56  ...          0.2060
0.07115
565         M        20.13  ...          0.2572
0.06637
566         M        16.60  ...          0.2218
0.07820
567         M        20.60  ...          0.4087
0.12400
568         B         7.76  ...          0.2871
0.07039

[569 rows x 31 columns]
```

```
# independent variables
x = df.drop('diagnosis',axis=1)
#dependent variables
y = df.diagnosis
```

```
x
```

```
     radius_mean  texture_mean  ...  symmetry_worst
fractal_dimension_worst
0          17.99         10.38  ...          0.4601
0.11890
1          20.57         17.77  ...          0.2750
0.08902
2          19.69         21.25  ...          0.3613
0.08758
3          11.42         20.38  ...          0.6638
0.17300
4          20.29         14.34  ...          0.2364
0.07678
..           ...           ...  ...             ...
...
564        21.56         22.39  ...          0.2060
0.07115
565        20.13         28.25  ...          0.2572
0.06637
566        16.60         28.08  ...          0.2218
0.07820
567        20.60         29.33  ...          0.4087
0.12400
568         7.76         24.54  ...          0.2871
0.07039

[569 rows x 30 columns]
```

```
y
```

```
0      M
1      M
2      M
3      M
4      M
      ..
564    M
565    M
566    M
567    M
568    B
Name: diagnosis, Length: 569, dtype: object
```

```
from sklearn.preprocessing import LabelEncoder
#creating the object
```

```
lb = LabelEncoder()
y = lb.fit_transform(y)

y

array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
1,
       1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1,
1,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1,
1,
       0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 1,
0,
       0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 0, 1, 0,
1,
       1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
0,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0,
0,
       0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1,
1,
       1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1,
1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0,
0,
       0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1,
       1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1,
1,
       1, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
1,
       1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1,
1,
       0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0,
0,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0,
1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0,
0,
       0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
1,
       0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
0,
       0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0,
0,
```

```
       0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0,
0,
       0, 0, 0, 0, 0, 1, 0, 1, 1, 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, 0])
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test =
train_test_split(x,y,test_size=0.3,random_state=40) #splitting data
```

```
x_train.shape
```

```
(398, 30)
```

```
y_train.shape
```

```
(398,)
```

```
type(x_train)
```

```
pandas.core.frame.DataFrame
```

```
#importing StandardScaler
from sklearn.preprocessing import StandardScaler
#creating object
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

```
#importing keras
import keras
#importing sequential module
from keras.models import Sequential
# import dense module for hidden layers
from keras.layers import Dense
#importing activation functions
from keras.layers import LeakyReLU,PReLU,ELU
from keras.layers import Dropout
```

```
x_train[0]
```

```
array([-0.67237593, -0.6548285 , -0.54716701, -0.65051465,
0.88959772,
        1.2724018 ,  0.94095804,  0.63012529,  3.08910957,
1.58508333,
       -0.32002697,  0.41352037, -0.30763556, -0.36134242, -
0.54624086,
        0.65324855,  0.44052498,  1.09028671,  4.2277962 ,
0.34911544,
       -0.53525507,  0.13001276, -0.47138666, -0.52042506,
0.27511374,
        1.04673817,  0.83214193,  1.08508176,  4.71125141,
1.15290927])
```

```
x_test[0]
```

```
array([-0.1165852 , -1.45933662, -0.175571  , -0.22141406, -
0.32299438,
       -0.81816145, -0.98506733, -0.78025606, -0.80757699, -
0.52332865,
       -0.69771763, -0.96687031, -0.73083854, -0.52316829, -
0.91482504,
       -1.00324823, -0.86935795, -0.83733158,  0.10205091, -
0.78880159,
       -0.34526031, -1.46077085, -0.40530037, -0.40011864, -
0.70426333,
       -0.99228097, -1.14659972, -0.89764163, -0.14546607, -
0.95260458])
```

```
#creating model
model = Sequential()

#first hidden layer
model.add(Dense(units=9,kernel_initializer='he_uniform',activation='re
lu',input_dim=30))
#second hidden layer
model.add(Dense(units=9,kernel_initializer='he_uniform',activation='re
lu'))
# last layer or output layer
model.add(Dense(units=1,kernel_initializer='glorot_uniform',activation
='sigmoid'))

#taking summary of layers
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_12 (Dense)            (None, 9)                 279

 dense_13 (Dense)            (None, 9)                 90

 dense_14 (Dense)            (None, 1)                 10

=================================================================
Total params: 379
Trainable params: 379
Non-trainable params: 0
_________________________________________________________________
```

```
#compiling the ANN
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['ac
curacy'])
```

```
 #fitting the ANN to the training set
model = model.fit(x_train,y_train,batch_size=100,epochs=100)
```

```
Epoch 1/100
4/4 [==============================] - 0s 5ms/step - loss: 0.8605 -
accuracy: 0.5477
Epoch 2/100
4/4 [==============================] - 0s 5ms/step - loss: 0.8030 -
accuracy: 0.5854
Epoch 3/100
4/4 [==============================] - 0s 4ms/step - loss: 0.7485 -
accuracy: 0.6307
Epoch 4/100
4/4 [==============================] - 0s 4ms/step - loss: 0.7017 -
accuracy: 0.6583
Epoch 5/100
4/4 [==============================] - 0s 4ms/step - loss: 0.6581 -
accuracy: 0.6784
Epoch 6/100
4/4 [==============================] - 0s 5ms/step - loss: 0.6164 -
accuracy: 0.7060
Epoch 7/100
4/4 [==============================] - 0s 4ms/step - loss: 0.5817 -
accuracy: 0.7437
Epoch 8/100
4/4 [==============================] - 0s 4ms/step - loss: 0.5467 -
accuracy: 0.7588
Epoch 9/100
4/4 [==============================] - 0s 3ms/step - loss: 0.5170 -
accuracy: 0.7739
Epoch 10/100
4/4 [==============================] - 0s 6ms/step - loss: 0.4897 -
accuracy: 0.7990
Epoch 11/100
4/4 [==============================] - 0s 4ms/step - loss: 0.4633 -
accuracy: 0.8216
Epoch 12/100
4/4 [==============================] - 0s 4ms/step - loss: 0.4405 -
accuracy: 0.8317
Epoch 13/100
4/4 [==============================] - 0s 4ms/step - loss: 0.4194 -
accuracy: 0.8467
Epoch 14/100
4/4 [==============================] - 0s 4ms/step - loss: 0.3985 -
accuracy: 0.8543
Epoch 15/100
4/4 [==============================] - 0s 4ms/step - loss: 0.3802 -
accuracy: 0.8568
Epoch 16/100
4/4 [==============================] - 0s 5ms/step - loss: 0.3629 -
```

```
accuracy: 0.8693
Epoch 17/100
4/4 [==============================] - 0s 4ms/step - loss: 0.3475 -
accuracy: 0.8844
Epoch 18/100
4/4 [==============================] - 0s 5ms/step - loss: 0.3318 -
accuracy: 0.8945
Epoch 19/100
4/4 [==============================] - 0s 4ms/step - loss: 0.3171 -
accuracy: 0.8970
Epoch 20/100
4/4 [==============================] - 0s 4ms/step - loss: 0.3043 -
accuracy: 0.9020
Epoch 21/100
4/4 [==============================] - 0s 4ms/step - loss: 0.2919 -
accuracy: 0.9045
Epoch 22/100
4/4 [==============================] - 0s 5ms/step - loss: 0.2801 -
accuracy: 0.9045
Epoch 23/100
4/4 [==============================] - 0s 5ms/step - loss: 0.2701 -
accuracy: 0.9121
Epoch 24/100
4/4 [==============================] - 0s 5ms/step - loss: 0.2597 -
accuracy: 0.9196
Epoch 25/100
4/4 [==============================] - 0s 5ms/step - loss: 0.2502 -
accuracy: 0.9221
Epoch 26/100
4/4 [==============================] - 0s 3ms/step - loss: 0.2417 -
accuracy: 0.9246
Epoch 27/100
4/4 [==============================] - 0s 4ms/step - loss: 0.2331 -
accuracy: 0.9246
Epoch 28/100
4/4 [==============================] - 0s 3ms/step - loss: 0.2248 -
accuracy: 0.9322
Epoch 29/100
4/4 [==============================] - 0s 3ms/step - loss: 0.2172 -
accuracy: 0.9322
Epoch 30/100
4/4 [==============================] - 0s 5ms/step - loss: 0.2101 -
accuracy: 0.9322
Epoch 31/100
4/4 [==============================] - 0s 5ms/step - loss: 0.2042 -
accuracy: 0.9322
Epoch 32/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1981 -
accuracy: 0.9347
Epoch 33/100
```

```
4/4 [==============================] - 0s 4ms/step - loss: 0.1925 -
accuracy: 0.9372
Epoch 34/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1870 -
accuracy: 0.9372
Epoch 35/100
4/4 [==============================] - 0s 4ms/step - loss: 0.1822 -
accuracy: 0.9372
Epoch 36/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1777 -
accuracy: 0.9372
Epoch 37/100
4/4 [==============================] - 0s 4ms/step - loss: 0.1737 -
accuracy: 0.9372
Epoch 38/100
4/4 [==============================] - 0s 4ms/step - loss: 0.1696 -
accuracy: 0.9372
Epoch 39/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1660 -
accuracy: 0.9372
Epoch 40/100
4/4 [==============================] - 0s 9ms/step - loss: 0.1623 -
accuracy: 0.9372
Epoch 41/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1586 -
accuracy: 0.9397
Epoch 42/100
4/4 [==============================] - 0s 4ms/step - loss: 0.1551 -
accuracy: 0.9397
Epoch 43/100
4/4 [==============================] - 0s 4ms/step - loss: 0.1515 -
accuracy: 0.9397
Epoch 44/100
4/4 [==============================] - 0s 4ms/step - loss: 0.1484 -
accuracy: 0.9422
Epoch 45/100
4/4 [==============================] - 0s 4ms/step - loss: 0.1451 -
accuracy: 0.9447
Epoch 46/100
4/4 [==============================] - 0s 4ms/step - loss: 0.1420 -
accuracy: 0.9447
Epoch 47/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1393 -
accuracy: 0.9472
Epoch 48/100
4/4 [==============================] - 0s 4ms/step - loss: 0.1365 -
accuracy: 0.9472
Epoch 49/100
4/4 [==============================] - 0s 4ms/step - loss: 0.1336 -
accuracy: 0.9497
```

```
Epoch 50/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1312 -
accuracy: 0.9497
Epoch 51/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1285 -
accuracy: 0.9548
Epoch 52/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1261 -
accuracy: 0.9548
Epoch 53/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1240 -
accuracy: 0.9548
Epoch 54/100
4/4 [==============================] - 0s 4ms/step - loss: 0.1217 -
accuracy: 0.9573
Epoch 55/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1195 -
accuracy: 0.9623
Epoch 56/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1174 -
accuracy: 0.9623
Epoch 57/100
4/4 [==============================] - 0s 4ms/step - loss: 0.1152 -
accuracy: 0.9623
Epoch 58/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1134 -
accuracy: 0.9648
Epoch 59/100
4/4 [==============================] - 0s 4ms/step - loss: 0.1114 -
accuracy: 0.9648
Epoch 60/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1096 -
accuracy: 0.9673
Epoch 61/100
4/4 [==============================] - 0s 4ms/step - loss: 0.1079 -
accuracy: 0.9673
Epoch 62/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1063 -
accuracy: 0.9698
Epoch 63/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1048 -
accuracy: 0.9698
Epoch 64/100
4/4 [==============================] - 0s 4ms/step - loss: 0.1033 -
accuracy: 0.9698
Epoch 65/100
4/4 [==============================] - 0s 4ms/step - loss: 0.1018 -
accuracy: 0.9749
Epoch 66/100
4/4 [==============================] - 0s 4ms/step - loss: 0.1005 -
```

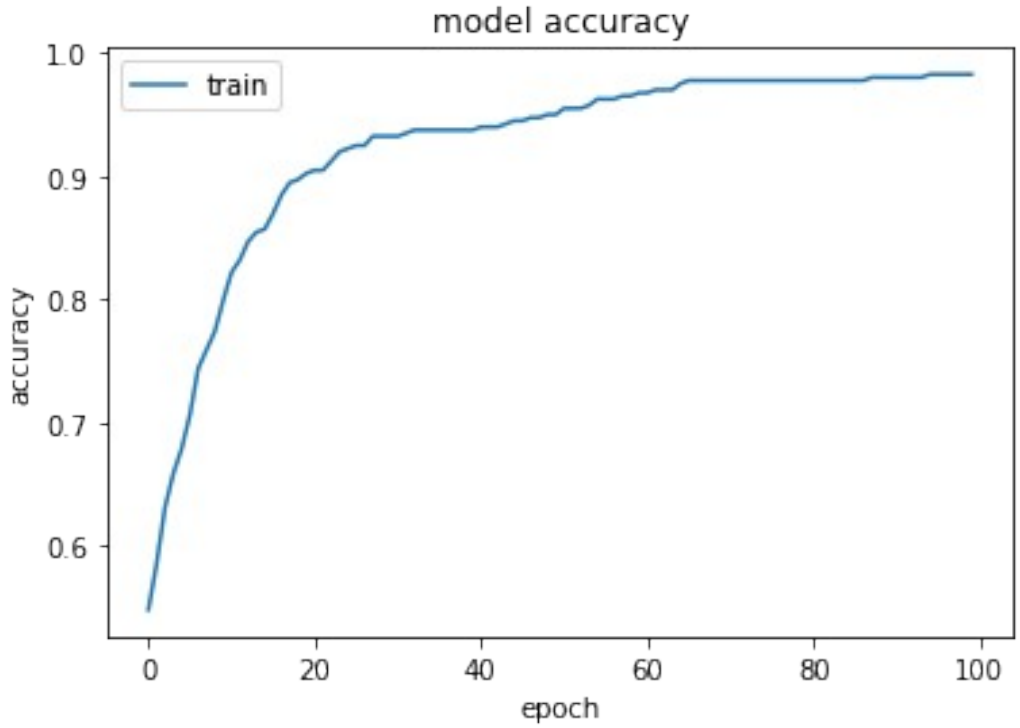
```
accuracy: 0.9774
Epoch 67/100
4/4 [==============================] - 0s 4ms/step - loss: 0.0991 -
accuracy: 0.9774
Epoch 68/100
4/4 [==============================] - 0s 4ms/step - loss: 0.0977 -
accuracy: 0.9774
Epoch 69/100
4/4 [==============================] - 0s 4ms/step - loss: 0.0964 -
accuracy: 0.9774
Epoch 70/100
4/4 [==============================] - 0s 5ms/step - loss: 0.0950 -
accuracy: 0.9774
Epoch 71/100
4/4 [==============================] - 0s 5ms/step - loss: 0.0939 -
accuracy: 0.9774
Epoch 72/100
4/4 [==============================] - 0s 5ms/step - loss: 0.0928 -
accuracy: 0.9774
Epoch 73/100
4/4 [==============================] - 0s 4ms/step - loss: 0.0918 -
accuracy: 0.9774
Epoch 74/100
4/4 [==============================] - 0s 4ms/step - loss: 0.0908 -
accuracy: 0.9774
Epoch 75/100
4/4 [==============================] - 0s 4ms/step - loss: 0.0898 -
accuracy: 0.9774
Epoch 76/100
4/4 [==============================] - 0s 4ms/step - loss: 0.0888 -
accuracy: 0.9774
Epoch 77/100
4/4 [==============================] - 0s 5ms/step - loss: 0.0880 -
accuracy: 0.9774
Epoch 78/100
4/4 [==============================] - 0s 4ms/step - loss: 0.0870 -
accuracy: 0.9774
Epoch 79/100
4/4 [==============================] - 0s 4ms/step - loss: 0.0862 -
accuracy: 0.9774
Epoch 80/100
4/4 [==============================] - 0s 7ms/step - loss: 0.0854 -
accuracy: 0.9774
Epoch 81/100
4/4 [==============================] - 0s 5ms/step - loss: 0.0845 -
accuracy: 0.9774
Epoch 82/100
4/4 [==============================] - 0s 6ms/step - loss: 0.0837 -
accuracy: 0.9774
Epoch 83/100
```

```
4/4 [==============================] - 0s 5ms/step - loss: 0.0829 -
accuracy: 0.9774
Epoch 84/100
4/4 [==============================] - 0s 4ms/step - loss: 0.0822 -
accuracy: 0.9774
Epoch 85/100
4/4 [==============================] - 0s 4ms/step - loss: 0.0814 -
accuracy: 0.9774
Epoch 86/100
4/4 [==============================] - 0s 5ms/step - loss: 0.0807 -
accuracy: 0.9774
Epoch 87/100
4/4 [==============================] - 0s 5ms/step - loss: 0.0800 -
accuracy: 0.9774
Epoch 88/100
4/4 [==============================] - 0s 4ms/step - loss: 0.0793 -
accuracy: 0.9799
Epoch 89/100
4/4 [==============================] - 0s 4ms/step - loss: 0.0786 -
accuracy: 0.9799
Epoch 90/100
4/4 [==============================] - 0s 4ms/step - loss: 0.0780 -
accuracy: 0.9799
Epoch 91/100
4/4 [==============================] - 0s 6ms/step - loss: 0.0773 -
accuracy: 0.9799
Epoch 92/100
4/4 [==============================] - 0s 6ms/step - loss: 0.0767 -
accuracy: 0.9799
Epoch 93/100
4/4 [==============================] - 0s 4ms/step - loss: 0.0761 -
accuracy: 0.9799
Epoch 94/100
4/4 [==============================] - 0s 4ms/step - loss: 0.0755 -
accuracy: 0.9799
Epoch 95/100
4/4 [==============================] - 0s 4ms/step - loss: 0.0749 -
accuracy: 0.9824
Epoch 96/100
4/4 [==============================] - 0s 5ms/step - loss: 0.0743 -
accuracy: 0.9824
Epoch 97/100
4/4 [==============================] - 0s 4ms/step - loss: 0.0738 -
accuracy: 0.9824
Epoch 98/100
4/4 [==============================] - 0s 5ms/step - loss: 0.0732 -
accuracy: 0.9824
Epoch 99/100
4/4 [==============================] - 0s 5ms/step - loss: 0.0727 -
accuracy: 0.9824
```

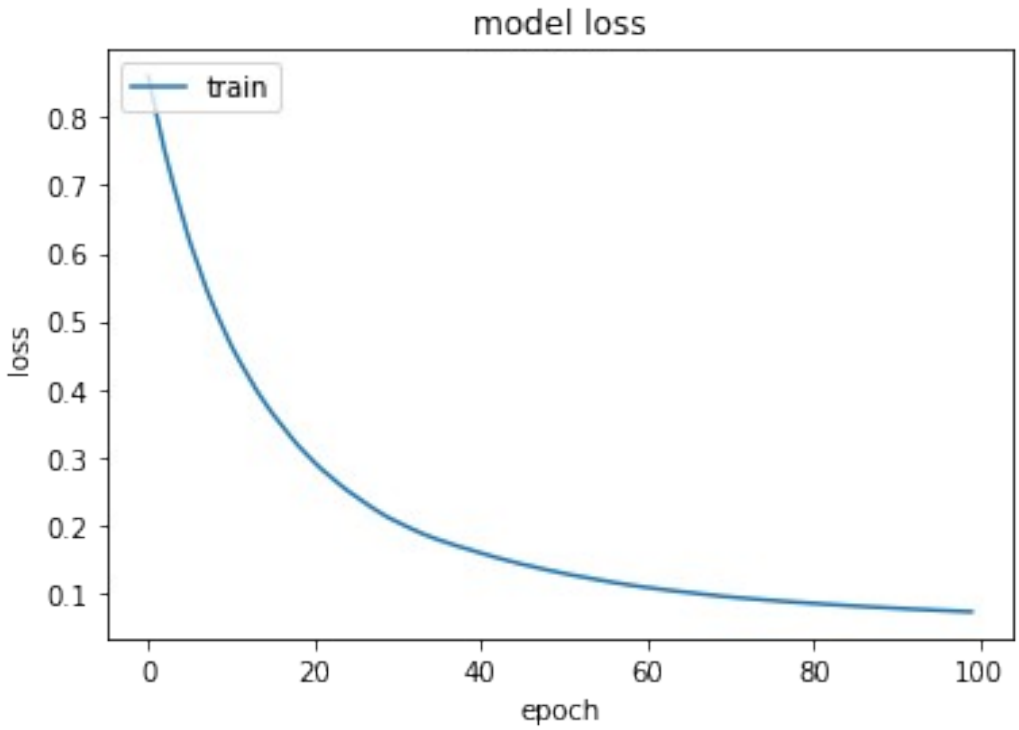
```
Epoch 100/100
4/4 [==============================] - 0s 5ms/step - loss: 0.0722 -
accuracy: 0.9824
```

```
# list all data in history
print(model.history.keys())
# summarize history for accuracy
plt.plot(model.history['accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

```
dict_keys(['loss', 'accuracy'])
```




```
# summarize history for loss
plt.plot(model.history['loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



RESULT:

The program executed successfully and obtained the output.

PROGRAM - 12

AIM:

Programs on convolutional neural network to classify images from any standard dataset in the public domain.

```
import tensorflow as tf
from tensorflow.keras import datasets,layers,models
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import classification_report
```

```
(x_train,y_train),(x_test,y_test) = datasets.cifar10.load_data()
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.g
170500096/170498071 [==============================] - 3s 0us/step
170508288/170498071 [==============================] - 3s 0us/step
```

```
x_train.shape
```

```
(50000, 32, 32, 3)
```

```
x_test.shape
```

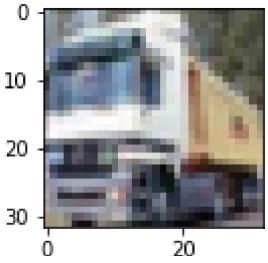
```
(10000, 32, 32, 3)
```

```
classes = ['airplane','automobile','bird','cat','deer','dog','frog','horse','ship
```

```
def plot_sample(x,y,index):
  plt.figure(figsize=(14,2))
  plt.imshow(x[index])
  plt.xlabel(y[index])
```

```
plot_sample(x_train,y_train,1)
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/text.py:1165: FutureWarnir
  if s != self._text:
```




```
#Normalize
x_train=x_train/255
x_test=x_test/255


#model
cnn=models.Sequential([
 #feature extraction
 layers.Conv2D(filters=32,activation='relu',kernel_size=(3,3),input_shape=(32,32,
 layers.MaxPooling2D((2,2)),

 #Classification
 layers.Flatten(),
 layers.Dense(64,activation='relu'),
 layers.Dense(10,activation='softmax')
])


cnn.compile(optimizer='adam',loss='sparse_categorical_crossentropy',metrics=['acc


cnn.fit(x_train,y_train,epochs=20)
```

```
Epoch 1/20
1563/1563 [==============================] - 33s 21ms/step - loss: 1.4955 -
Epoch 2/20
1563/1563 [==============================] - 32s 21ms/step - loss: 1.2080 -
Epoch 3/20
1563/1563 [==============================] - 32s 21ms/step - loss: 1.1052 -
Epoch 4/20
1563/1563 [==============================] - 33s 21ms/step - loss: 1.0290 -
Epoch 5/20
1563/1563 [==============================] - 31s 20ms/step - loss: 0.9726 -
Epoch 6/20
1563/1563 [==============================] - 31s 20ms/step - loss: 0.9200 -
Epoch 7/20
1563/1563 [==============================] - 32s 21ms/step - loss: 0.8766 -
Epoch 8/20
1563/1563 [==============================] - 32s 21ms/step - loss: 0.8384 -
Epoch 9/20
```

```
1563/1563 [==============================] - 32s 20ms/step - loss: 0.8063 -
Epoch 10/20
1563/1563 [==============================] - 32s 21ms/step - loss: 0.7773 -
Epoch 11/20
1563/1563 [==============================] - 32s 20ms/step - loss: 0.7443 -
Epoch 12/20
1563/1563 [==============================] - 35s 22ms/step - loss: 0.7181 -
Epoch 13/20
1563/1563 [==============================] - 33s 21ms/step - loss: 0.6937 -
Epoch 14/20
1563/1563 [==============================] - 33s 21ms/step - loss: 0.6653 -
Epoch 15/20
1563/1563 [==============================] - 34s 22ms/step - loss: 0.6452 -
Epoch 16/20
1563/1563 [==============================] - 35s 22ms/step - loss: 0.6167 -
Epoch 17/20
1563/1563 [==============================] - 33s 21ms/step - loss: 0.5945 -
Epoch 18/20
1563/1563 [==============================] - 33s 21ms/step - loss: 0.5793 -
Epoch 19/20
1563/1563 [==============================] - 33s 21ms/step - loss: 0.5546 -
Epoch 20/20
1563/1563 [==============================] - 33s 21ms/step - loss: 0.5316 -
<keras.callbacks.History at 0x7f999ade9b90>
```

```
y_pred=cnn.predict(x_test)
```

```
cnn.evaluate(x_test,y_test)
```

```
313/313 [==============================] - 2s 7ms/step - loss: 1.3166 - accu
[1.3166470527648926, 0.6133999824523926]
```

```
y_test=y_test.reshape(-1,)
y_pred=cnn.predict(x_test)
```

```
y_classes=[np.argmax(element) for element in y_pred]
print("Classification report:\n",classification_report(y_test,y_classes))
```

```
Classification report:
               precision    recall  f1-score   support

           0       0.64      0.65      0.65      1000
           1       0.82      0.65      0.72      1000
           2       0.48      0.50      0.49      1000
           3       0.41      0.46      0.43      1000
           4       0.64      0.45      0.53      1000
```

```
              5       0.49      0.53      0.51      1000
              6       0.74      0.68      0.70      1000
              7       0.65      0.69      0.67      1000
              8       0.70      0.77      0.73      1000
              9       0.65      0.76      0.70      1000

       accuracy                           0.61     10000
      macro avg       0.62      0.61      0.61     10000
   weighted avg       0.62      0.61      0.61     10000
```

RESULT:

The program executed successfully and obtained the output.