```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import models
from tensorflow.keras import layers
from keras.models import Sequential
from keras.layers import Dense,Flatten
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd


# loading the dataset
(xtrain,ytrain),(xtest,ytest)=keras.datasets.mnist.load_data()


xtrain.shape   # 60000-images data having 28*28 pixel each
```

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

```
xtrain  # showing the images in pixel form
```

```
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, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]]], dtype=uint8)
```

```
xtrain[0]   # displays pixels of 1st image having 28 rows and 28 columns
```

```
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,   3,
  18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,
 253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,
 253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253, 253,
 253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,
 205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,
  90,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139, 253,
 190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190,
 253,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35,
 241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
  81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,  45, 186, 253, 253, 150,  27,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,  16,  93, 252, 253, 187,   0,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,   0,   0,   0, 249, 253, 249,  64,   0,   0,   0,   0,   0,
   0,   0],
[  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
   0,  46, 130, 183, 253, 253, 207,   2,   0,   0,   0,   0,   0,
   0,   0],
```

```
xtest.shape
```

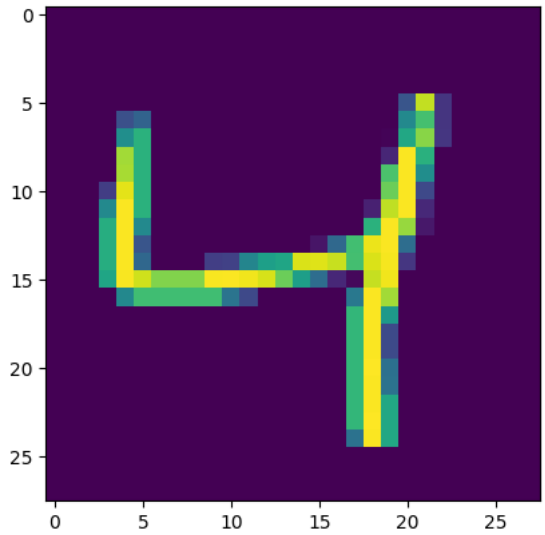
```
(10000, 28, 28)
```

```
ytrain # It has labels showing which images has which number.for eg-1st image has number 5 and so on.
```

```
array([5, 0, 4, ..., 5, 6, 8], dtype=uint8)
```

```
plt.imshow(xtrain[2])
```

```
<matplotlib.image.AxesImage at 0x7f53c9df76d0>
```




```
# making the values of pixels between 0 and 1 by dividing it by 255
# to make conversion faster and effient classification.
xtrain=xtrain/255
xtest=xtest/255
```

```
model=Sequential()
model.add(Flatten(input_shape=(28,28)))
model.add(Dense(128,activation='relu'))
model.add(Dense(10,activation='softmax'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 128)               100480

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(loss='sparse_categorical_crossentropy',optimizer='adam')
```

```
model.fit(xtrain,ytrain,epochs=10,validation_split=0.2)
```

```
Epoch 1/10
1500/1500 [==============================] - 9s 5ms/step - loss: 0.2879 - val_loss: 0.1622
Epoch 2/10
1500/1500 [==============================] - 7s 4ms/step - loss: 0.1291 - val_loss: 0.1196
Epoch 3/10
1500/1500 [==============================] - 7s 5ms/step - loss: 0.0861 - val_loss: 0.1199
Epoch 4/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0657 - val_loss: 0.0928
Epoch 5/10
1500/1500 [==============================] - 9s 6ms/step - loss: 0.0501 - val_loss: 0.0842
Epoch 6/10
1500/1500 [==============================] - 8s 6ms/step - loss: 0.0391 - val_loss: 0.0910
Epoch 7/10
1500/1500 [==============================] - 7s 4ms/step - loss: 0.0327 - val_loss: 0.0856
Epoch 8/10
1500/1500 [==============================] - 8s 5ms/step - loss: 0.0258 - val_loss: 0.0825
Epoch 9/10
1500/1500 [==============================] - 7s 4ms/step - loss: 0.0205 - val_loss: 0.1041
Epoch 10/10
1500/1500 [==============================] - 9s 6ms/step - loss: 0.0167 - val_loss: 0.0997
<keras.callbacks.History at 0x7f53a951c7c0>
```

```
yprob=model.predict(xtest)  # it gives probability of a number between 0 to 9 for each image.
# for eg=it shows 2.39013742e-09 for number 0, shows 1.11755799e-10 for 1 for image1 and so on.
```

```
313/313 [==============================] - 1s 2ms/step
```

```
yprob
```

```
array([[2.39013742e-09, 1.11755799e-10, 9.74204610e-08, ...,
        9.99962151e-01, 4.45395765e-09, 1.10615952e-06],
       [1.78723078e-11, 1.28480707e-07, 9.99999821e-01, ...,
        5.74466596e-15, 3.38614203e-09, 5.87625190e-16],
       [5.69186000e-07, 9.99325216e-01, 9.72087291e-05, ...,
        3.58759164e-04, 1.56880415e-04, 4.79745985e-08],
       ...,
       [4.45055628e-17, 1.28302600e-15, 1.95706685e-16, ...,
        1.75010437e-10, 1.84384452e-10, 3.73865333e-07],
       [4.21595668e-16, 3.07799660e-15, 2.43570253e-16, ...,
        1.73322446e-12, 2.64826006e-09, 1.14054299e-14],
       [4.07153244e-08, 1.02798216e-13, 9.94845877e-08, ...,
        3.64297177e-12, 2.19490856e-10, 1.89057321e-08]], dtype=float32)
```

```
ypred=yprob.argmax(axis=1) # it finds max value(max probability) for the particular number.
```

```
ypred
```

```
array([7, 2, 1, ..., 4, 5, 6])
```

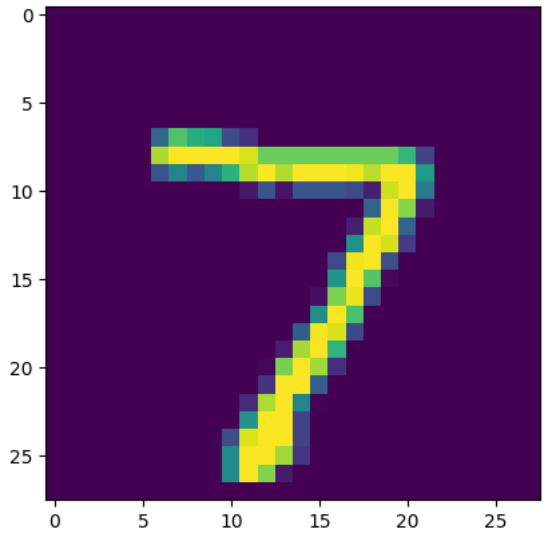
```
from sklearn.metrics import accuracy_score
```

```
accuracy_score(ytest,ypred)
```

```
0.9732
```

```
plt.imshow(xtest[0])
```

```
<matplotlib.image.AxesImage at 0x7f53a951fcd0>
```




```
# checking the value if it predicted correctly or not
model.predict(xtest[0].reshape(1,28,28)).argmax(axis=1)
```

```
1/1 [==============================] - 0s 38ms/step
array([7])
```