```
import tensorflow as tf
from tensorflow import keras
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
```

```
from tensorflow.keras.datasets import mnist
```

```
(trainX, trainY), (testX, testY) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnis
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
len (trainX)
```

```
60000
```

```
len(testX)
```

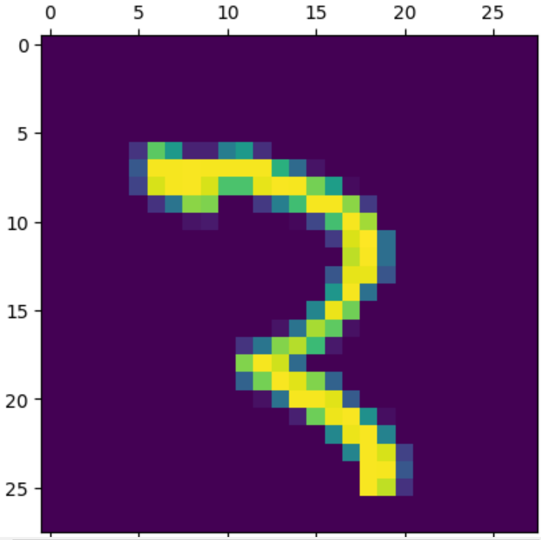
```
10000
```

```
trainX.shape
```

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

```
plt.matshow(trainX[500])
```

```
<matplotlib.image.AxesImage at 0x7afad7be7c40>
```



```
#normaliseimages
trainX=trainX/255
testX=testX/255
```

```
trainX[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, 0.53333333, 0.68627451, 0.10196078,
          0.65098039, 1.        , 0.96862745, 0.49803922, 0.        ,
          0.        , 0.        , 0.        ],
         [0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.11764706, 0.14117647,
          0.36862745, 0.60392157, 0.66666667, 0.99215686, 0.99215686,
          0.99215686, 0.99215686, 0.99215686, 0.88235294, 0.6745098 ,
          0.99215686, 0.94901961, 0.76470588, 0.25098039, 0.        ,
          0.        , 0.        , 0.        ],
         [0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.19215686, 0.93333333, 0.99215686,
          0.99215686, 0.99215686, 0.99215686, 0.99215686, 0.99215686,
          0.99215686, 0.99215686, 0.98431373, 0.36470588, 0.32156863,
          0.32156863, 0.21960784, 0.15294118, 0.        , 0.        ,
          0.        , 0.        , 0.        ],
         [0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.07058824, 0.85882353, 0.99215686,
          0.99215686, 0.99215686, 0.99215686, 0.99215686, 0.77647059,
          0.71372549, 0.96862745, 0.94509804, 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        ],
         [0.        , 0.        , 0.        , 0.        , 0.        ,
          0.        , 0.        , 0.        , 0.31372549, 0.61176471,
          0.41960784, 0.99215686, 0.99215686, 0.80392157, 0.04313725,
          0.        , 0.16862745, 0.60392157, 0.        , 0.        ,
```

```
model=keras.Sequential([
    keras.layers.Flatten(input_shape=(28,28)),
    keras.layers.Dense(128,activation='relu'),
    keras.layers.Dense(10,activation='softmax')
])
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_1 (Flatten)         (None, 784)               0

 dense_1 (Dense)             (None, 128)               100480

 dense_2 (Dense)             (None, 10)                1290

=================================================================
Total params: 101770 (397.54 KB)
Trainable params: 101770 (397.54 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
model.compile(optimizer="sgd",
              loss="sparse_categorical_crossentropy",
              metrics=['accuracy'])


history=model.fit(trainX, trainY, validation_data=(testX, testY), epochs=10)
```

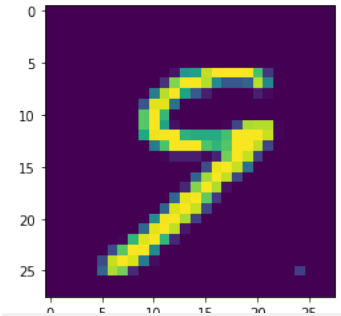
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-7a960ee5a744> in <cell line: 1>()
----> 1 history=model.fit(trainX, trainY, validation_data=(testX, testY), epochs=10)

NameError: name 'model' is not defined
```

```
test_loss, test_acc=model.evaluate(testX, testY)
print("Loss=%.3f" %test_loss)
print("Accuracy=%.3f" %test_acc)
```

```
258/313 [=======================>......] - ETA: 0s - loss: 0.1751 - accuracy: 0.948826
313/313 [==============================] - 0s 590us/step - loss: 0.1653 - accuracy: 0.
Loss=0.165
Accuracy=0.952
```

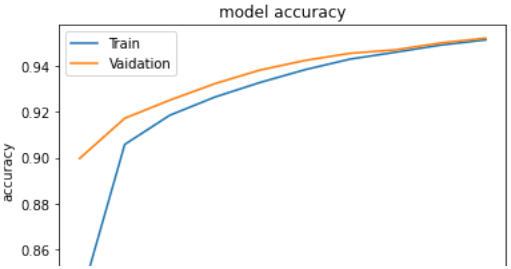
```
n=random.randint(0,9999)
plt.imshow(testX[n])
plt.show()
```




```
predicted_value=model.predict(testX)
print("Handwritten number in the image is=%d" %np.argmax(predicted_value[n]))
```

```
313/313 [==============================] - 0s 510us/step
Handwritten number in the image is=9
```

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('eopch')
plt.legend(['Train', 'Vaidation'], loc='upper left')
plt.show()
```




```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Training loss and Accuracy')
plt.ylabel('accuracy/Loss')
plt.xlabel('eopch')
plt.legend(['accuracy', 'val_accuaracy', 'loss', 'val_loss'])
plt.show()
```