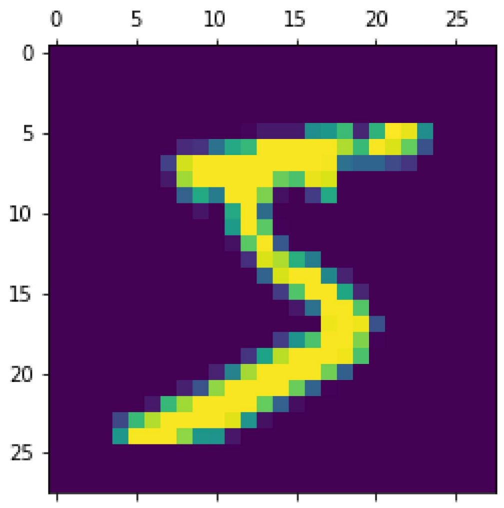
In [1]:
```
import tensorflow as tf
from tensorflow import keras
```

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
%matplotlib inline
```

In [3]:
```
mnist=tf.keras.datasets.mnist
(x_train,y_train),(x_test,y_test)=mnist.load_data()
```

In [4]:
```
plt.matshow(x_train[0])
```

Out[4]:
```
<matplotlib.image.AxesImage at 0x1db0793a160>
```



In [5]:
```
x_train=x_train/255
x_test=x_test/255
```

In [6]:
```
x_train[0]
```

```
         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,
```

In [7]:
```
model=keras.Sequential([keras.layers.Flatten(input_shape=(28,28)),
                       keras.layers.Dense(128,activation='relu'),
                       keras.layers.Dense(10,activation='softmax')])
```

In [8]:
```
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
_________________________________________________________________
```

In [9]:
```
model.compile(optimizer='sgd',loss='sparse_categorical_crossentropy',metrics=['ac
```

In [10]:
```
history=model.fit(x_train,y_train,validation_data=(x_test,y_test),epochs=10)
```

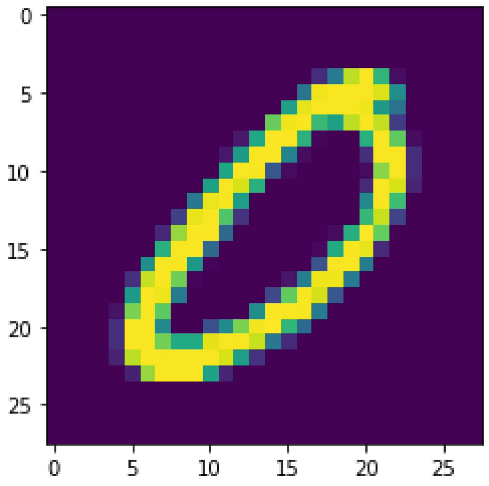
```
Epoch 1/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.6424 - accur
acy: 0.8357 - val_loss: 0.3572 - val_accuracy: 0.9029
Epoch 2/10
1875/1875 [==============================] - 10s 5ms/step - loss: 0.3373 - accu
racy: 0.9054 - val_loss: 0.2941 - val_accuracy: 0.9165
Epoch 3/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2889 - accur
acy: 0.9184 - val_loss: 0.2622 - val_accuracy: 0.9267
Epoch 4/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2590 - accur
acy: 0.9272 - val_loss: 0.2377 - val_accuracy: 0.9347
Epoch 5/10
1875/1875 [==============================] - 10s 5ms/step - loss: 0.2364 - accu
racy: 0.9341 - val_loss: 0.2216 - val_accuracy: 0.9388
Epoch 6/10
1875/1875 [==============================] - 10s 5ms/step - loss: 0.2181 - accu
racy: 0.9388 - val_loss: 0.2058 - val_accuracy: 0.9442
Epoch 7/10
1875/1875 [==============================] - 10s 5ms/step - loss: 0.2025 - accu
racy: 0.9436 - val_loss: 0.1924 - val_accuracy: 0.9474
Epoch 8/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.1891 - accur
acy: 0.9472 - val_loss: 0.1777 - val_accuracy: 0.9504
Epoch 9/10
1875/1875 [==============================] - 9s 5ms/step - loss: 0.1770 - accur
acy: 0.9507 - val_loss: 0.1690 - val_accuracy: 0.9530
Epoch 10/10
1875/1875 [==============================] - 10s 5ms/step - loss: 0.1668 - accu
racy: 0.9537 - val_loss: 0.1599 - val_accuracy: 0.9552
```

In [11]:
```
test_loss,test_acc=model.evaluate(x_test,y_test)
print("Loss=%.3f" %test_loss)
print("Accuracy=%0.3f" %test_acc)
```

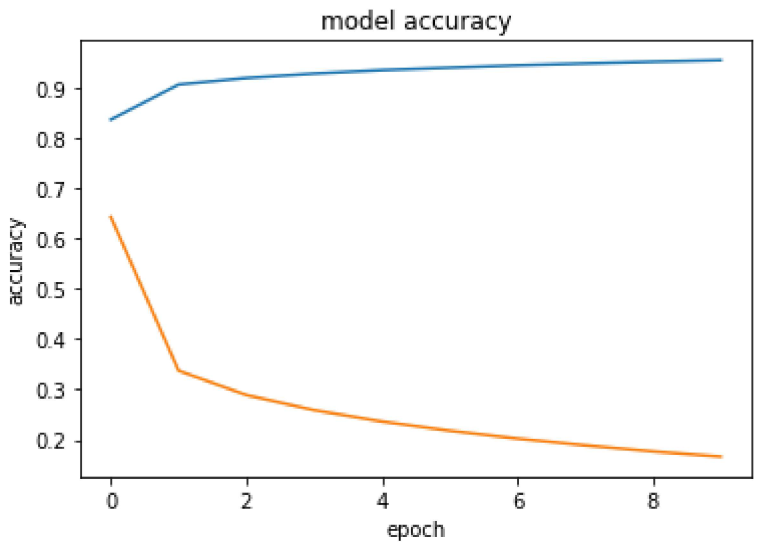
```
313/313 [==============================] - 1s 2ms/step - loss: 0.1599 - accurac
y: 0.9552
Loss=0.160
Accuracy=0.955
```

In [12]:
```
n=random.randint(0,9999)
plt.imshow(x_test[n])
plt.show
```

Out[12]:
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [27]:
```
plt.plot(history.history['accuracy'],label="accuracy")
plt.plot(history.history['loss'],label="loss")
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.show()
```



In [14]:
```
test_predict=model.predict(x_test)
test_predict_labels=np.argmax(test_predict,axis=1)
confusion_matrix=tf.math.confusion_matrix(labels=y_test,predictions=test_predict_
print(confusion_matrix)
```

```
tf.Tensor(
[[ 961    0    2    1    0    4    8    2    1    1]
 [    0 1115    3    2    0    1    3    2    9    0]
 [    7    1  982    9    6    0    6    7   13    1]
 [    0    0   12  965    0    8    0   11   11    3]
 [    1    2    4    0  938    0   10    4    3   20]
 [    8    2    0   21    2  828   12    2   11    6]
 [    8    3    2    1    7    7  925    0    5    0]
 [    0    8   20    5    4    1    0  976    2   12]
 [    4    2    4   10    5    5    9   10  923    2]
 [    8    8    2   12   22    2    1    9    6  939]], shape=(10, 10), dtype=in
t32)
```

In [ ]: