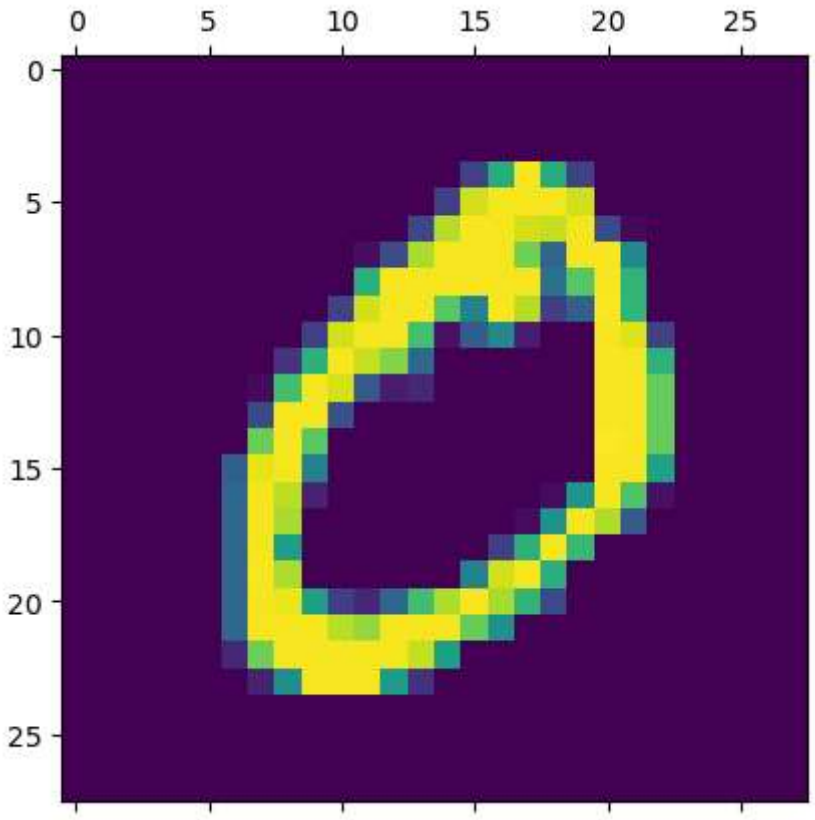
In [ ]:
```
#importing necessary libraries
import tensorflow as tf
from tensorflow import keras
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
%matplotlib inline
```

In [3]:
```
#import dataset and split into train and test data
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

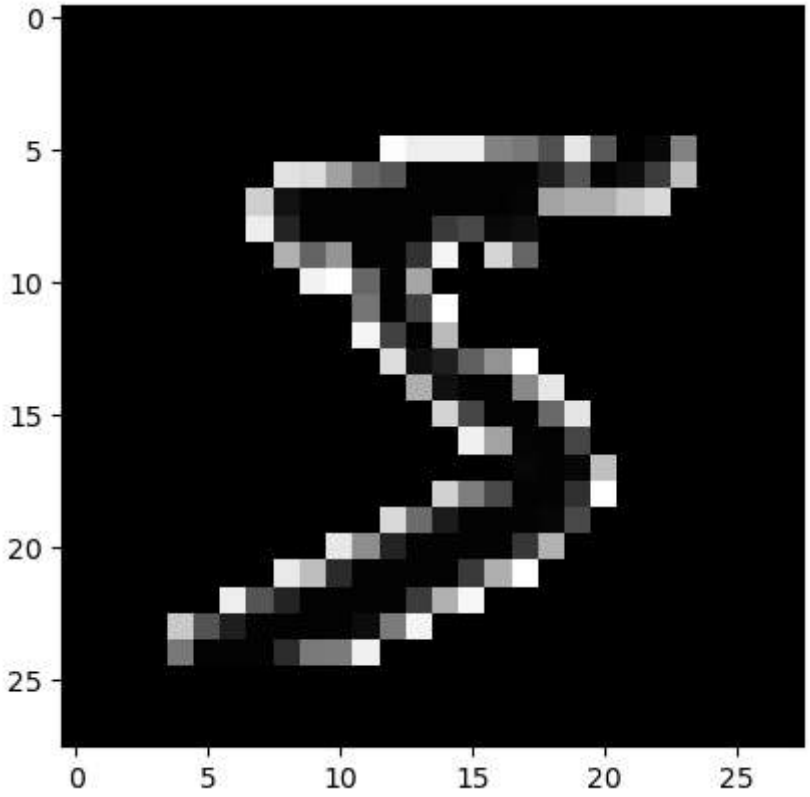
In [4]:
```
plt.matshow(x_train[1])
```

Out[4]:
```
<matplotlib.image.AxesImage at 0x2318f9c2788>
```



In [5]:
```
plt.imshow(-x_train[0], cmap="gray")
```

Out[5]:
```
<matplotlib.image.AxesImage at 0x2318f7af388>
```



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

In [6]:
```
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 (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 [7]:
```
model.compile(optimizer="sgd",
loss="sparse_categorical_crossentropy",
metrics=['accuracy'])
```

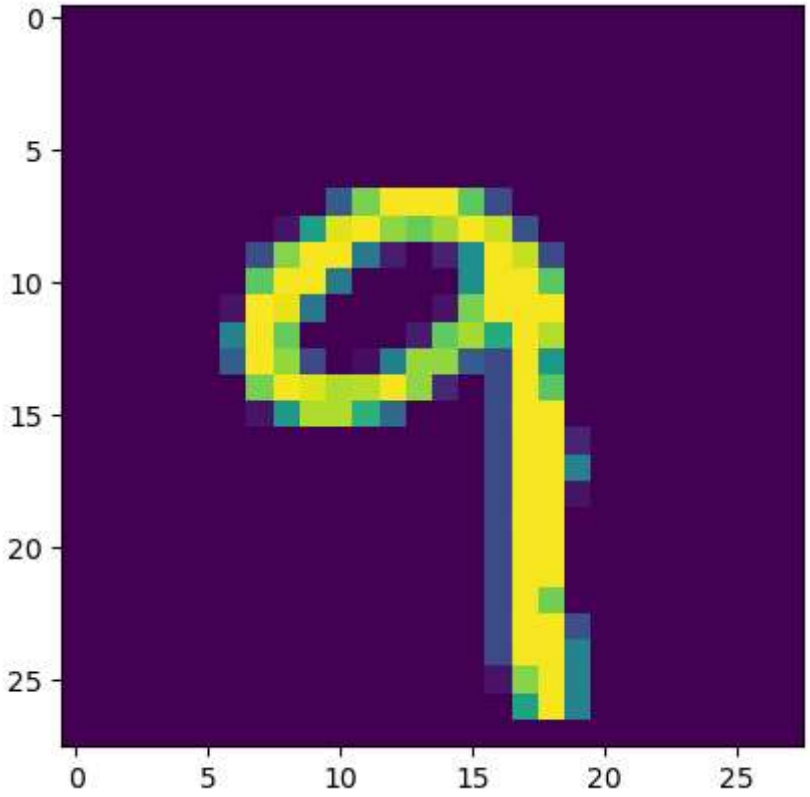
In [8]:
```
history=model.fit(x_train,
y_train,validation_data=(x_test,y_test),epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 1s 619us/step - loss: 0.6385 - accuracy:
0.8383 - val_loss: 0.3554 - val_accuracy: 0.9044
Epoch 2/10
1875/1875 [==============================] - 1s 552us/step - loss: 0.3383 - accuracy:
0.9051 - val_loss: 0.2977 - val_accuracy: 0.9176
Epoch 3/10
1875/1875 [==============================] - 1s 546us/step - loss: 0.2920 - accuracy:
0.9176 - val_loss: 0.2663 - val_accuracy: 0.9275
Epoch 4/10
1875/1875 [==============================] - 1s 545us/step - loss: 0.2623 - accuracy:
0.9269 - val_loss: 0.2438 - val_accuracy: 0.9319
Epoch 5/10
1875/1875 [==============================] - 1s 547us/step - loss: 0.2399 - accuracy:
0.9331 - val_loss: 0.2238 - val_accuracy: 0.9377
Epoch 6/10
1875/1875 [==============================] - 1s 658us/step - loss: 0.2211 - accuracy:
0.9385 - val_loss: 0.2086 - val_accuracy: 0.9399
Epoch 7/10
1875/1875 [==============================] - 1s 639us/step - loss: 0.2054 - accuracy:
0.9432 - val_loss: 0.1947 - val_accuracy: 0.9440
Epoch 8/10
1875/1875 [==============================] - 1s 627us/step - loss: 0.1919 - accuracy:
0.9467 - val_loss: 0.1847 - val_accuracy: 0.9461
Epoch 9/10
1875/1875 [==============================] - 1s 556us/step - loss: 0.1801 - accuracy:
0.9500 - val_loss: 0.1736 - val_accuracy: 0.9500
Epoch 10/10
1875/1875 [==============================] - 1s 552us/step - loss: 0.1695 - accuracy:
0.9535 - val_loss: 0.1656 - val_accuracy: 0.9521
```

In [9]:
```
test_loss,test_acc=model.evaluate(x_test,y_test)
print("Loss=%.3f" %test_loss)
print("Accuracy=%.3f" %test_acc)
```

```
313/313 [==============================] - 0s 424us/step - loss: 0.1656 - accuracy:
0.9521
Loss=0.166
Accuracy=0.952
```

In [10]:
```
n=random.randint(0,9999)
plt.imshow(x_test[n])
plt.show()
```



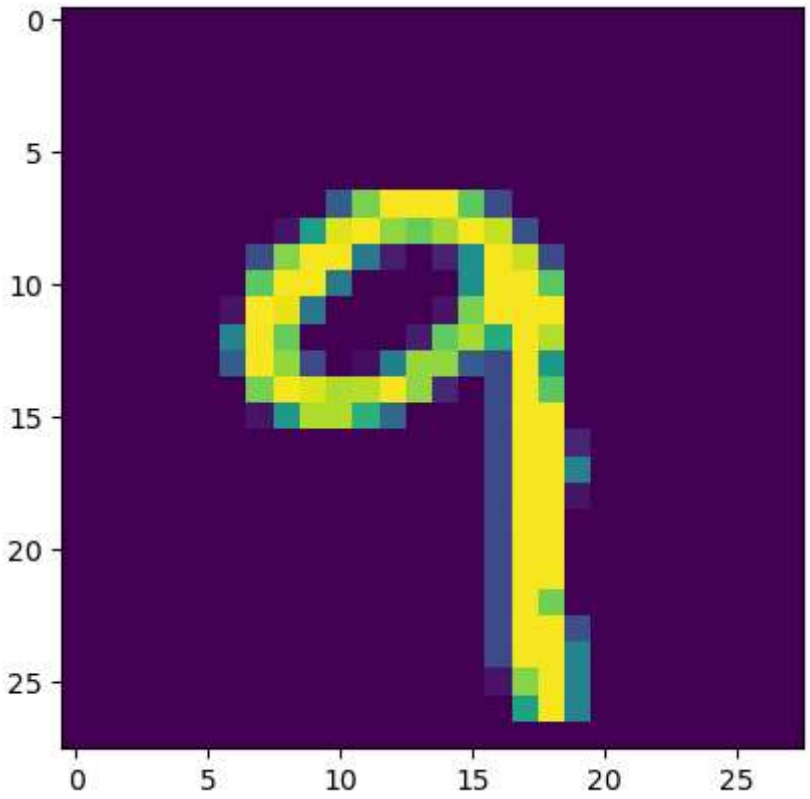
In [11]:
```
x_train
```

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., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       ...,

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]]])
```

In [12]:
```
x_test
```

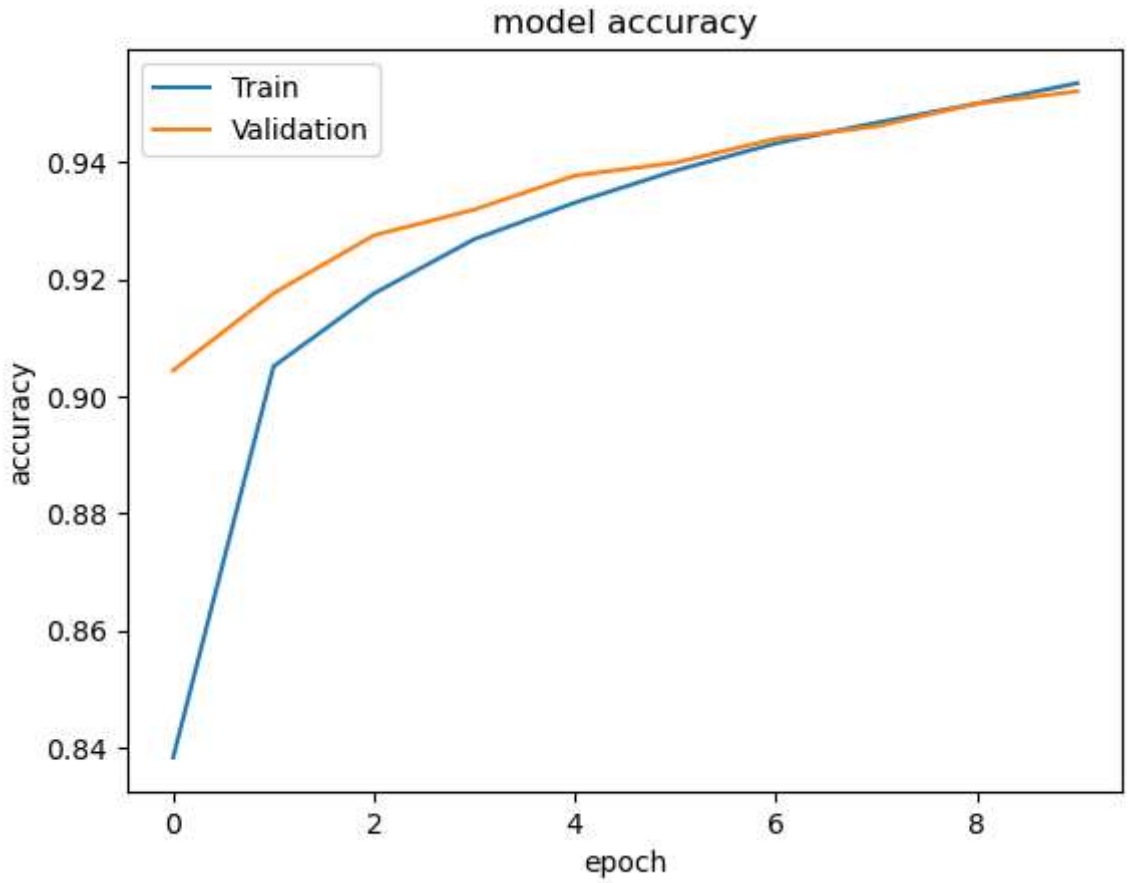
Out[12]:
```
array([[[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[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.]]])
```

In [13]:
```
predicted_value=model.predict(x_test)
plt.imshow(x_test[n])
plt.show()

print(predicted_value[n])
```

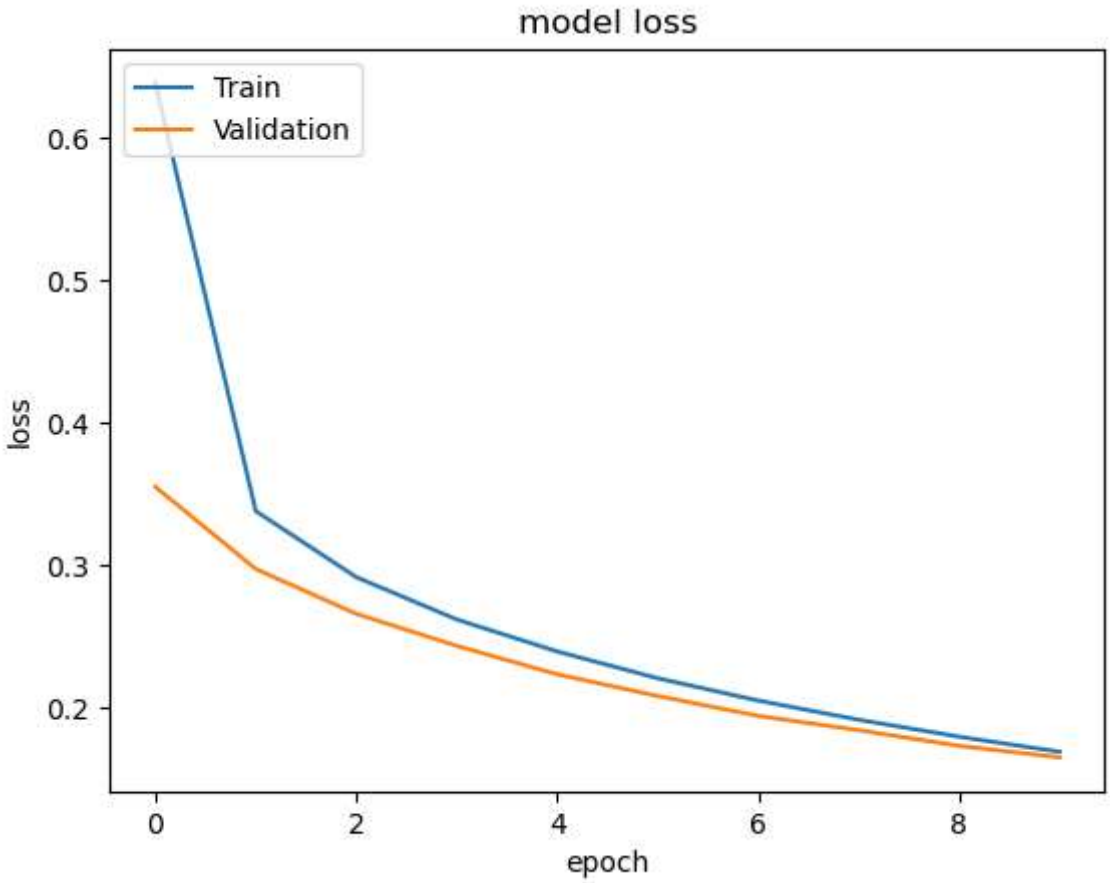



```
[7.2151641e-05 3.4558122e-06 3.3534434e-05 4.9630189e-03 2.8357599e-03
 1.7946344e-03 4.8126030e-06 1.5905222e-02 6.7955634e-04 9.7370785e-01]
```

In [14]:
```
# history.history()
history.history.keys()
# dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])

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



In [15]:
```
# history.history()
history.history.keys()
# dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```



In [ ]: