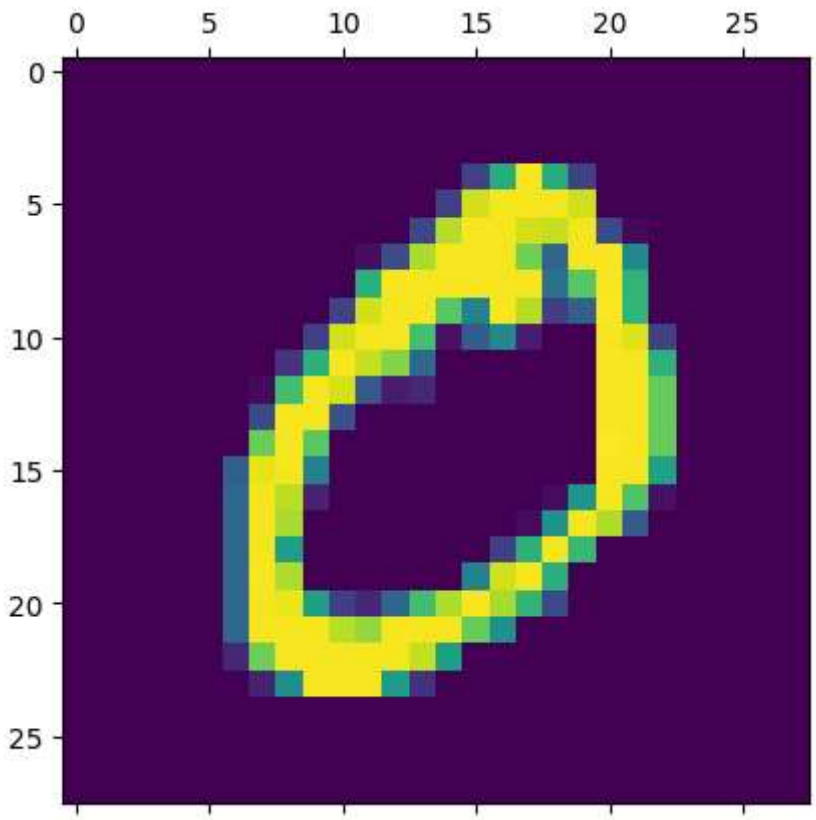
In [1]:
```
#Name: Ankita Durgude
#Roll No: 18
#Batch: B1
#RMDSSOE BE IT
```

In [2]:
```
#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()
```

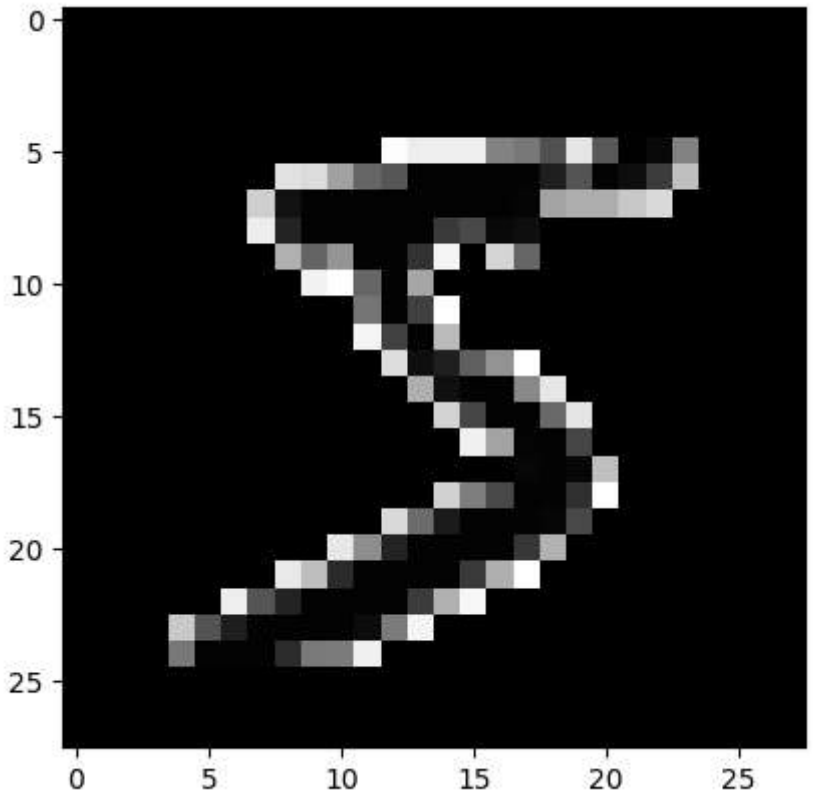
In [4]:
```
plt.matshow(x_train[1])
```

Out[4]: <matplotlib.image.AxesImage at 0x21bbb470a90>



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

Out[5]: <matplotlib.image.AxesImage at 0x21bbe9cea70>



In [6]:
```
x_train = x_train / 255
x_test = x_test / 255
```

In [7]:
```
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 [8]:
```
model.compile(optimizer="sgd",
loss="sparse_categorical_crossentropy",
metrics=['accuracy'])
```

In [9]:
```
history=model.fit(x_train,
```

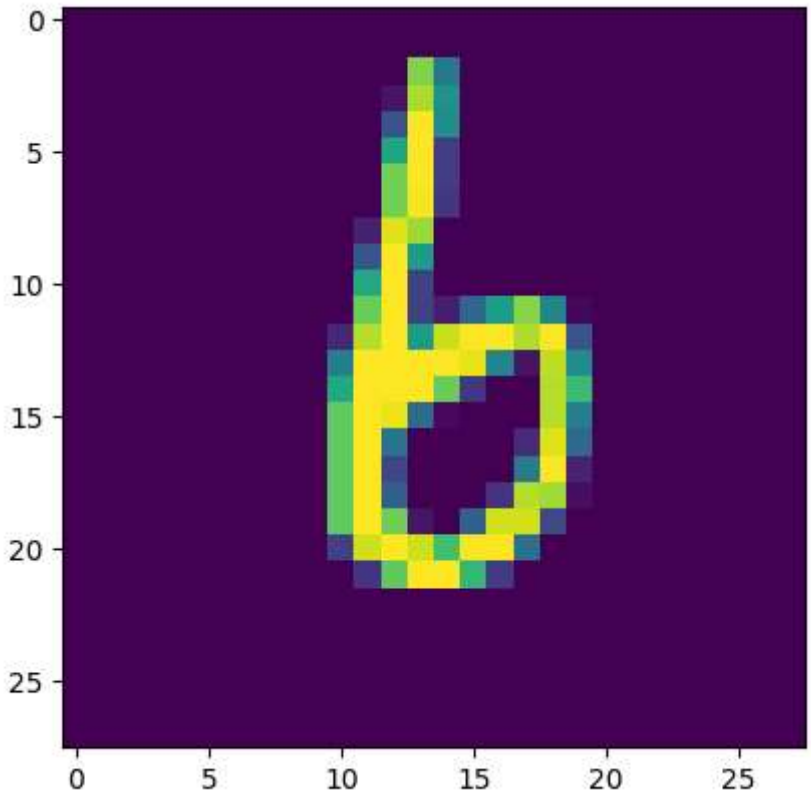
```
y_train,validation_data=(x_test,y_test),epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 10s 4ms/step - loss: 0.6452 - accuracy:
0.8372 - val_loss: 0.3578 - val_accuracy: 0.9021
Epoch 2/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.3374 - accuracy:
0.9061 - val_loss: 0.2964 - val_accuracy: 0.9172
Epoch 3/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2912 - accuracy:
0.9183 - val_loss: 0.2671 - val_accuracy: 0.9269
Epoch 4/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2622 - accuracy:
0.9259 - val_loss: 0.2422 - val_accuracy: 0.9323
Epoch 5/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2397 - accuracy:
0.9325 - val_loss: 0.2265 - val_accuracy: 0.9366
Epoch 6/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2211 - accuracy:
0.9375 - val_loss: 0.2078 - val_accuracy: 0.9399
Epoch 7/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.2052 - accuracy:
0.9424 - val_loss: 0.1966 - val_accuracy: 0.9433
Epoch 8/10
1875/1875 [==============================] - 7s 4ms/step - loss: 0.1918 - accuracy:
0.9460 - val_loss: 0.1853 - val_accuracy: 0.9457
Epoch 9/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.1798 - accuracy:
0.9495 - val_loss: 0.1748 - val_accuracy: 0.9492
Epoch 10/10
1875/1875 [==============================] - 8s 4ms/step - loss: 0.1696 - accuracy:
0.9526 - val_loss: 0.1664 - val_accuracy: 0.9507
```

In [10]:
```
test_loss,test_acc=model.evaluate(x_test,y_test)
print("Loss=%.3f" %test_loss)
print("Accuracy=%.3f" %test_acc)
```

```
313/313 [==============================] - 1s 4ms/step - loss: 0.1664 - accuracy: 0.9
507
Loss=0.166
Accuracy=0.951
```

In [11]:
```
n=random.randint(0,9999)
plt.imshow(x_test[n])
plt.show()
```



In [12]:
```
x_train
```

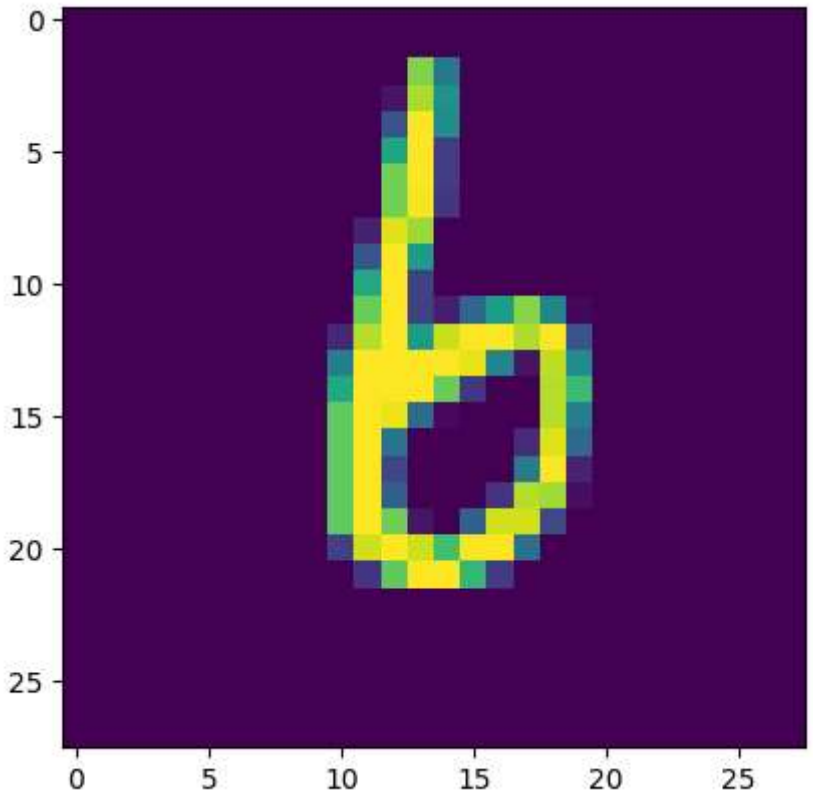
Out[12]:
```
array([[[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[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]:
```
x_test
```

Out[13]:
```
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.]]])
```

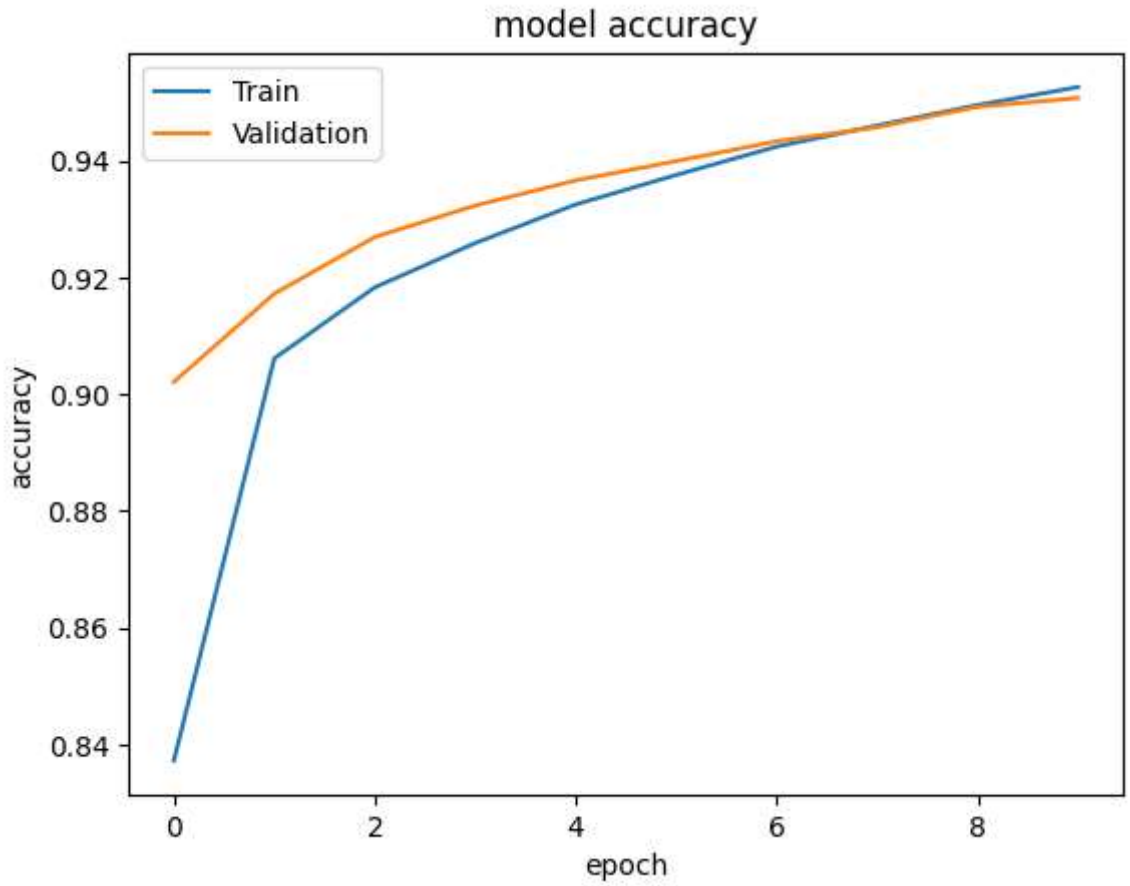
In [14]:
```
predicted_value=model.predict(x_test)
plt.imshow(x_test[n])
plt.show()

print(predicted_value[n])
```

```
313/313 [==============================] - 1s 3ms/step
```

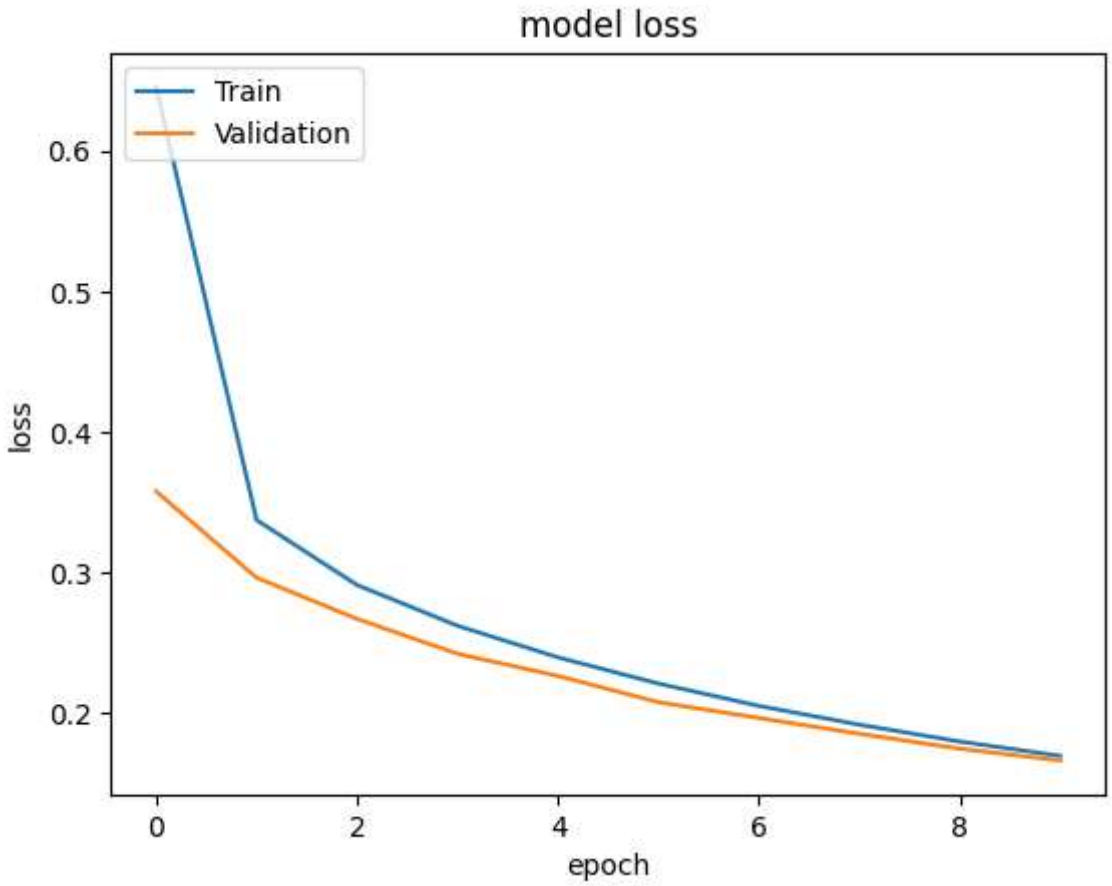



```
[7.0845248e-04 1.5189237e-04 3.0130744e-03 2.7047314e-03 2.0089766e-03
 9.5815780e-03 9.7662354e-01 1.2254562e-05 4.6167811e-03 5.7875580e-04]
```

In [15]:
```
# 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 [16]:
```
# 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 [ ]: