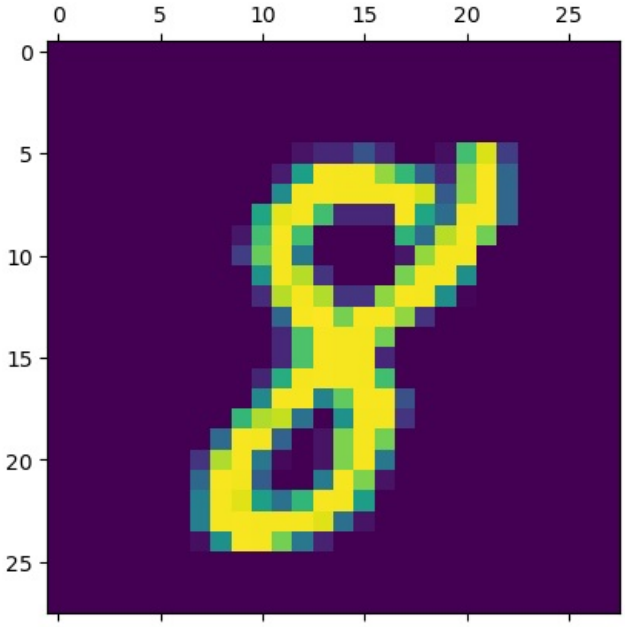
In [17]:
```
#importing necessary libraries
import tensorflow as tf
from tensorflow import keras
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
%matplotlib inline
```

In [18]:
```
#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()
```

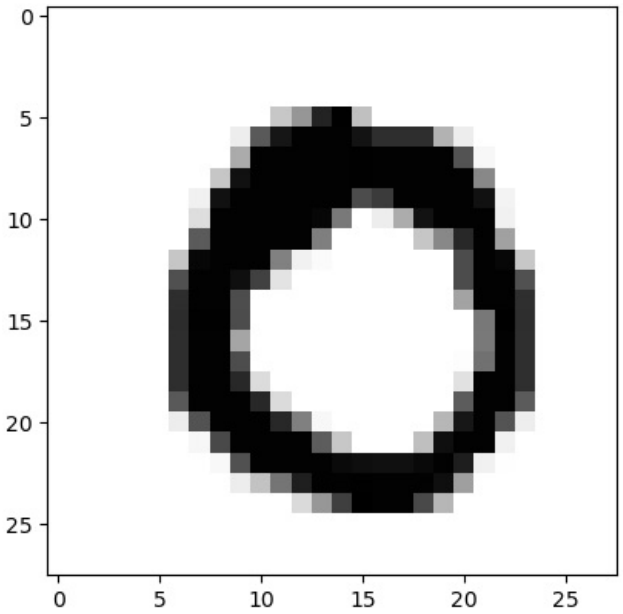
In [55]:
```
plt.matshow(x_train[97])
```

Out[55]:
```
<matplotlib.image.AxesImage at 0x288a21b5390>
```



In [56]:
```
plt.imshow(-x_train[56], cmap="gray")
```

Out[56]:
```
<matplotlib.image.AxesImage at 0x288a2365390>
```



In [58]:
```
x_train = x_train / 255
x_test = x_test / 255
# to bring the array in a specific range
print(x_train.min())
print(x_train.max())
```

```
0.0
2.365044290627876e-10
```

In [31]:
```
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_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_1 (Flatten)         (None, 784)               0

 dense_2 (Dense)             (None, 128)               100480

 dense_3 (Dense)             (None, 10)                1290

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

In [32]:
```
model.compile(optimizer="sgd",
loss="sparse_categorical_crossentropy",
metrics=['accuracy'])
```

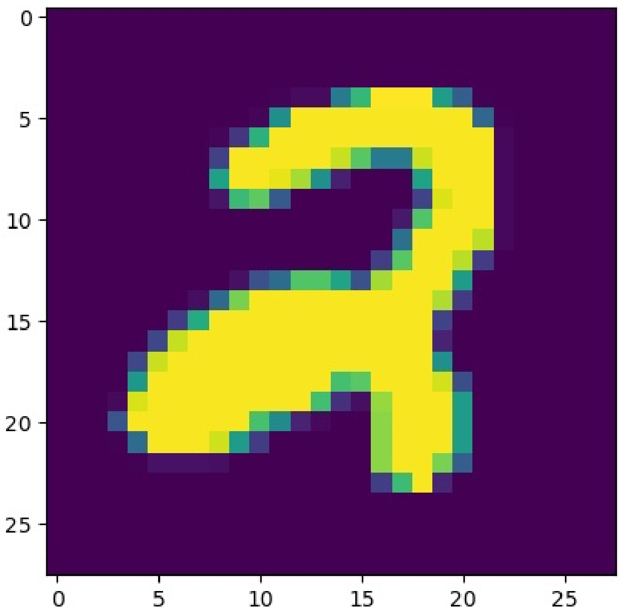
In [33]:
```
history=model.fit(x_train,
y_train,validation_data=(x_test,y_test),epochs=10)
```

```
Epoch 1/10
1875/1875 [==============================] - 10s 5ms/step - loss: 2.3003 - accuracy: 0.1128 - val_loss: 2.2995 -
val_accuracy: 0.1135
Epoch 2/10
1875/1875 [==============================] - 9s 5ms/step - loss: 2.2996 - accuracy: 0.1124 - val_loss: 2.2992 -
val_accuracy: 0.1135
Epoch 3/10
1875/1875 [==============================] - 9s 5ms/step - loss: 2.2992 - accuracy: 0.1124 - val_loss: 2.2987 -
val_accuracy: 0.1135
Epoch 4/10
1875/1875 [==============================] - 9s 5ms/step - loss: 2.2988 - accuracy: 0.1124 - val_loss: 2.2983 -
val_accuracy: 0.1135
Epoch 5/10
1875/1875 [==============================] - 9s 5ms/step - loss: 2.2984 - accuracy: 0.1124 - val_loss: 2.2978 -
val_accuracy: 0.1135
Epoch 6/10
1875/1875 [==============================] - 9s 5ms/step - loss: 2.2980 - accuracy: 0.1124 - val_loss: 2.2973 -
val_accuracy: 0.1135
Epoch 7/10
1875/1875 [==============================] - 9s 5ms/step - loss: 2.2975 - accuracy: 0.1124 - val_loss: 2.2968 -
val_accuracy: 0.1135
Epoch 8/10
1875/1875 [==============================] - 9s 5ms/step - loss: 2.2970 - accuracy: 0.1124 - val_loss: 2.2963 -
val_accuracy: 0.1135
Epoch 9/10
1875/1875 [==============================] - 9s 5ms/step - loss: 2.2965 - accuracy: 0.1124 - val_loss: 2.2959 -
val_accuracy: 0.1135
Epoch 10/10
1875/1875 [==============================] - 9s 5ms/step - loss: 2.2960 - accuracy: 0.1124 - val_loss: 2.2953 -
val_accuracy: 0.1135
```

In [43]:
```
test_loss,test_acc=model.evaluate(x_test,y_test)
print("Loss=%.3f" %test_loss)
print("Accuracy=%.3f" %test_acc)
```

```
313/313 [==============================] - 2s 5ms/step - loss: 2.3011 - accuracy: 0.1135
Loss=2.301
Accuracy=0.113
```

In [44]:
```
n=random.randint(0,9999)
plt.imshow(x_test[n])
plt.show()
```



In [48]:
```python
x_train[:5]
```

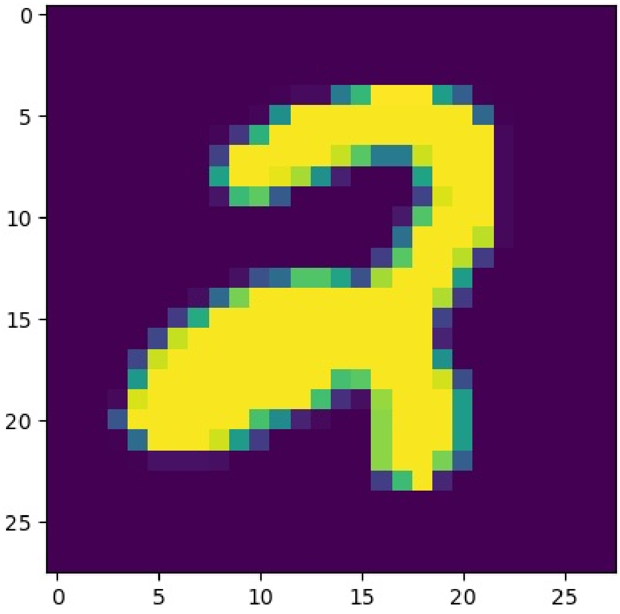
Out[48]:
```
array([[[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[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.]]])
```

In [50]:
```python
x_test[:5]
```

```
Out[50]:  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.]]])
```

```
In [53]: predicted_value=model.predict(x_test)
plt.imshow(x_test[n])
plt.show()

print(predicted_value[n])
```

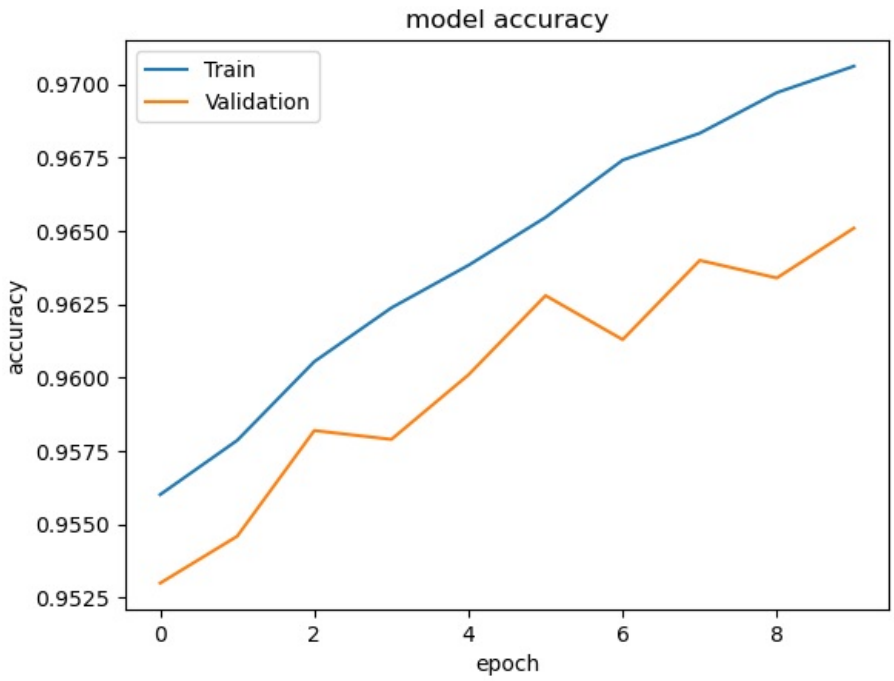
```
313/313 [==============================] - 1s 4ms/step
```




```
[0.10003445 0.11241449 0.09926774 0.10102703 0.09765565 0.08916723
 0.1017401  0.1027891  0.09555817 0.100346  ]
```

```
In [28]: # 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 [29]:
```
# 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()
```