In [42]:

```
#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 [43]:

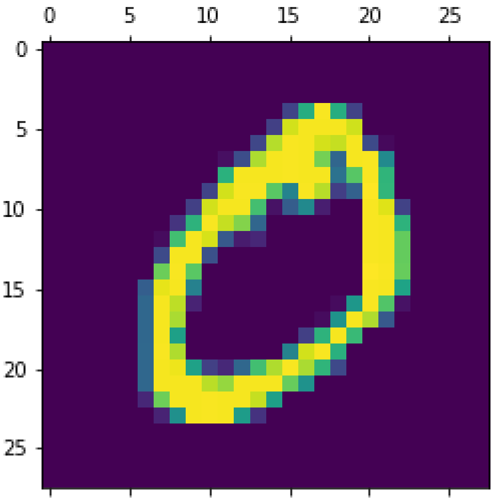
```
#import dataset and split into train and test data
mnist = tf.keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

In [44]:

```
plt.matshow(x_train[1])
```

Out[44]:

```
<matplotlib.image.AxesImage at 0x20c029b7070>
```

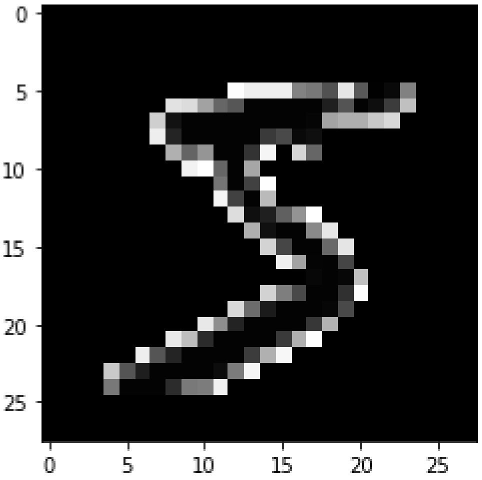


In [45]:

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

Out[45]:

```
<matplotlib.image.AxesImage at 0x20c027207c0>
```



In [46]:

```
x_train = x_train / 255
x_test = x_test / 255
```

In [47]:

```
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_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten_3 (Flatten)         (None, 784)               0

 dense_6 (Dense)             (None, 128)               100480

 dense_7 (Dense)             (None, 10)                1290

=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
_________________________________________________________________
```

In [48]:

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

In [49]:

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

```
Epoch 1/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.6438 - accuracy:
0.8396 - val_loss: 0.3600 - val_accuracy: 0.9020
Epoch 2/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.3398 - accuracy:
0.9045 - val_loss: 0.2939 - val_accuracy: 0.9196
Epoch 3/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2904 - accuracy:
0.9183 - val_loss: 0.2622 - val_accuracy: 0.9250
Epoch 4/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2594 - accuracy:
0.9272 - val_loss: 0.2400 - val_accuracy: 0.9323
Epoch 5/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2360 - accuracy:
0.9341 - val_loss: 0.2192 - val_accuracy: 0.9390
Epoch 6/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2174 - accuracy:
0.9393 - val_loss: 0.2042 - val_accuracy: 0.9409
Epoch 7/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.2018 - accuracy:
0.9433 - val_loss: 0.1895 - val_accuracy: 0.9464
Epoch 8/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.1886 - accuracy:
0.9473 - val_loss: 0.1801 - val_accuracy: 0.9481
Epoch 9/10
1875/1875 [==============================] - 3s 1ms/step - loss: 0.1771 - accuracy:
0.9502 - val_loss: 0.1701 - val_accuracy: 0.9521
Epoch 10/10
1875/1875 [==============================] - 2s 1ms/step - loss: 0.1669 - accuracy:
0.9534 - val_loss: 0.1615 - val_accuracy: 0.9537
```

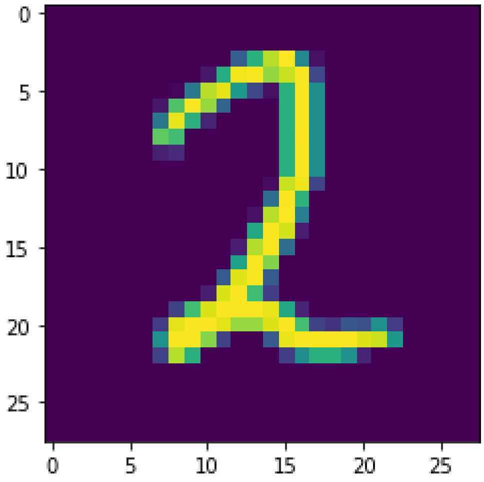
In [50]:

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

```
313/313 [==============================] - 0s 961us/step - loss: 0.1615 - accuracy:
0.9537
Loss=0.162
Accuracy=0.954
```

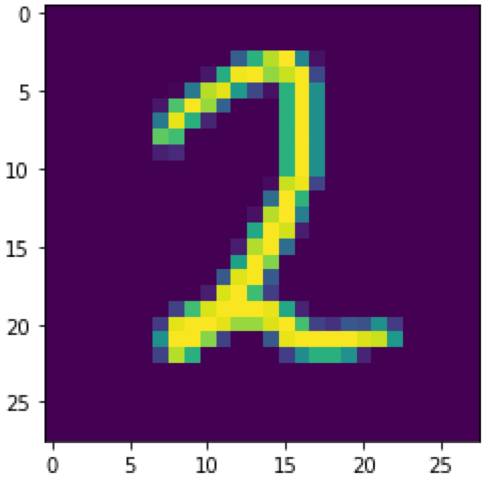
In [51]:

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



In [52]:

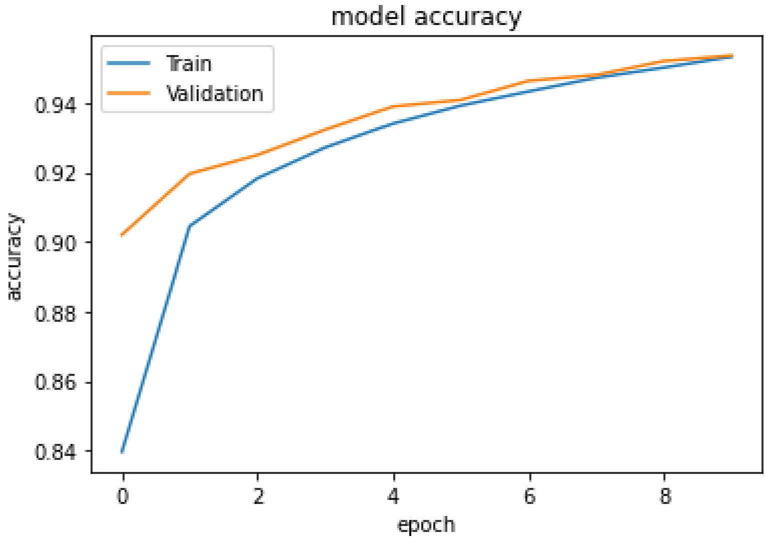
```
predicted_value=model.predict(x_test)
plt.imshow(x_test[n])
plt.show()
print(predicted_value[n])
```



```
[1.5068307e-06 9.8045049e-03 9.8612612e-01 3.5523567e-03 2.6645974e-08
 7.9984216e-05 9.4329873e-05 1.4903831e-05 3.2636689e-04 1.7516204e-08]
```

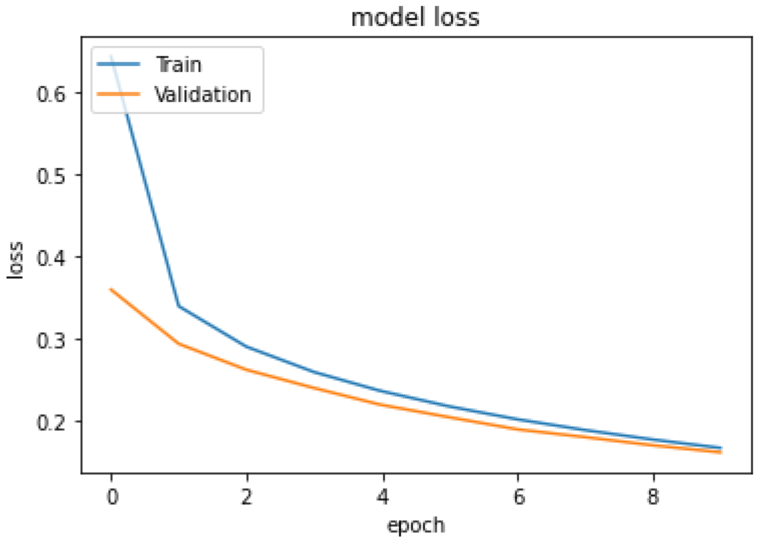
In [53]:

```
# 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 [54]:

```
# 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 [ ]: