In [18]:
```
import tensorflow
from tensorflow import keras
from tensorflow.keras import Sequential, Model
from tensorflow.keras.layers import Dense,Flatten
```

In [36]:
```
(X_train,y_train),(X_test,y_test) = keras.datasets.mnist.load_data()
```

In [37]:
```
X_train.shape
```

Out[37]:
```
(60000, 28, 28)
```

In [38]:
```
X_train
```

Out[38]:
```
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]]], dtype=uint8)
```

In [39]:
```
X_test
```

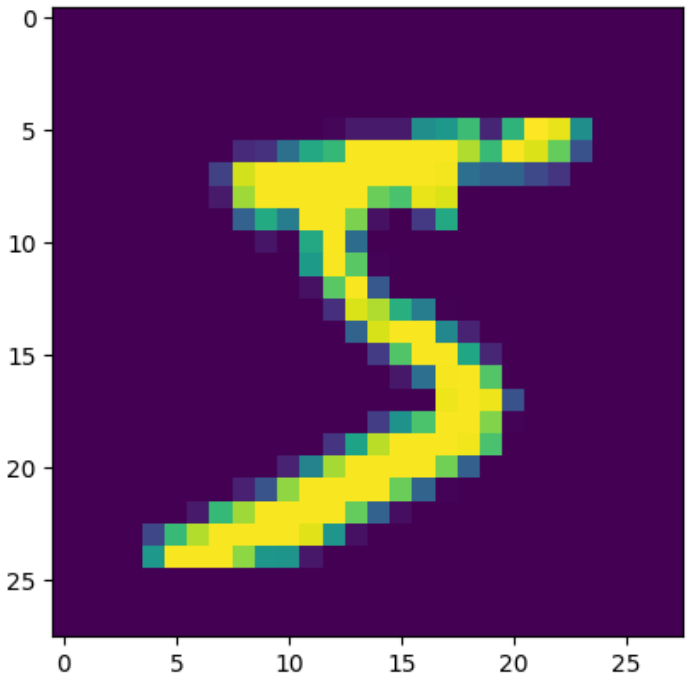
Out[39]:
```
array([[[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       [[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]]], dtype=uint8)
```

In [44]:
```
y_train
```

Out[44]:
```
array([5, 0, 4, ..., 5, 6, 8], dtype=uint8)
```

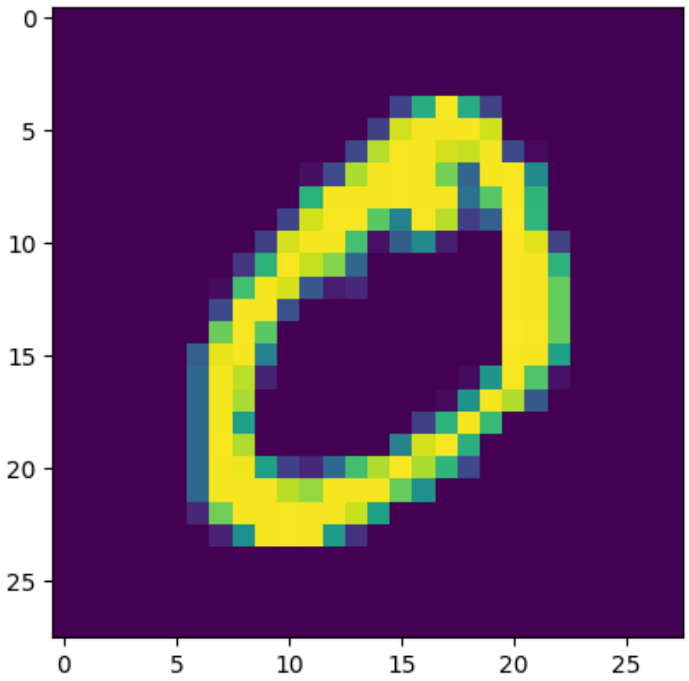
In [7]:
```
import matplotlib.pyplot as plt
plt.imshow(X_train[0])
```

Out[7]:
```
<matplotlib.image.AxesImage at 0x1ba047b9dc0>
```



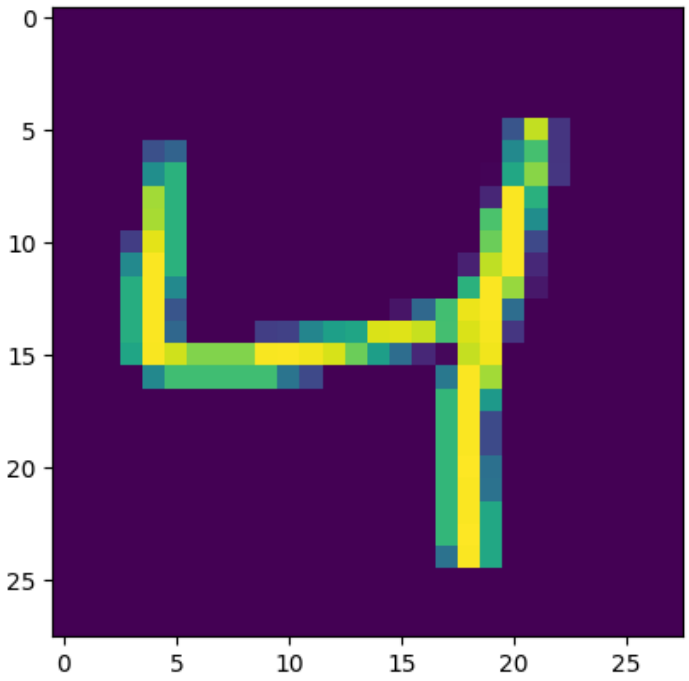
In [8]:
```
import matplotlib.pyplot as plt
plt.imshow(X_train[1])
```

Out[8]:
```
<matplotlib.image.AxesImage at 0x1ba050a68e0>
```



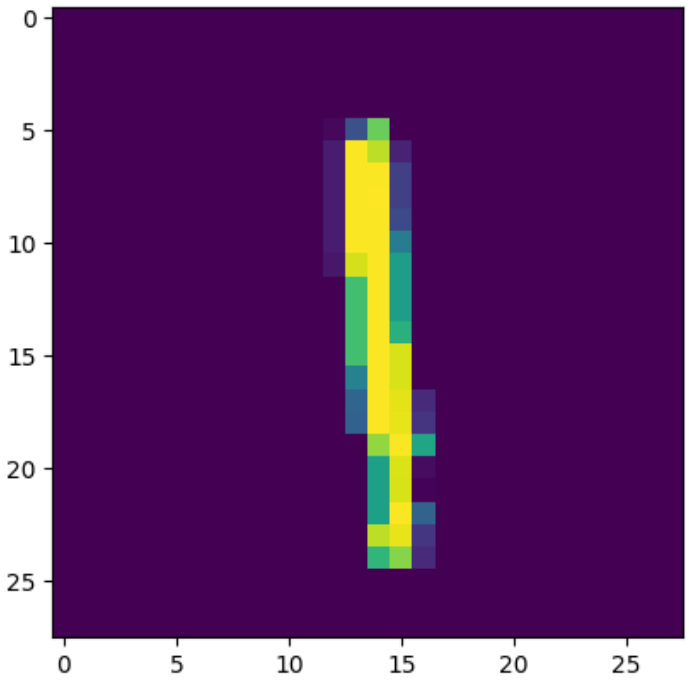
In [9]:
```
import matplotlib.pyplot as plt
plt.imshow(X_train[2])
```

Out[9]: <matplotlib.image.AxesImage at 0x1ba74e21e50>



In [10]:
```
import matplotlib.pyplot as plt
plt.imshow(X_train[8])
```

Out[10]: <matplotlib.image.AxesImage at 0x1ba04935ee0>



In [11]:
```
X_train[0]
```

```
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,   3,
                  18,  18,  18, 126, 136, 175,  26, 166, 255, 247, 127,   0,   0,
                   0,   0],
                [  0,   0,   0,   0,   0,   0,   0,   0,  30,  36,  94, 154, 170,
                 253, 253, 253, 253, 253, 225, 172, 253, 242, 195,  64,   0,   0,
                   0,   0],
                [  0,   0,   0,   0,   0,   0,   0,  49, 238, 253, 253, 253, 253,
                 253, 253, 253, 253, 251,  93,  82,  82,  56,  39,   0,   0,   0,
                   0,   0],
                [  0,   0,   0,   0,   0,   0,   0,  18, 219, 253, 253, 253, 253,
                 253, 198, 182, 247, 241,   0,   0,   0,   0,   0,   0,   0,   0,
                   0,   0],
                [  0,   0,   0,   0,   0,   0,   0,   0,  80, 156, 107, 253, 253,
                 205,  11,   0,  43, 154,   0,   0,   0,   0,   0,   0,   0,   0,
                   0,   0],
                [  0,   0,   0,   0,   0,   0,   0,   0,   0,  14,   1, 154, 253,
                  90,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                   0,   0],
                [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0, 139, 253,
                 190,   2,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                   0,   0],
                [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  11, 190,
                 253,  70,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                   0,   0],
                [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  35,
                 241, 225, 160, 108,   1,   0,   0,   0,   0,   0,   0,   0,   0,
                   0,   0],
                [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                  81, 240, 253, 253, 119,  25,   0,   0,   0,   0,   0,   0,   0,
                   0,   0],
                [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                   0,  45, 186, 253, 253, 150,  27,   0,   0,   0,   0,   0,   0,
                   0,   0],
                [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                   0,   0,  16,  93, 252, 253, 187,   0,   0,   0,   0,   0,   0,
                   0,   0],
                [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                   0,   0,   0,   0, 249, 253, 249,  64,   0,   0,   0,   0,   0,
                   0,   0],
                [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                   0,  46, 130, 183, 253, 253, 207,   2,   0,   0,   0,   0,   0,
                   0,   0],
                [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  39,
                 148, 229, 253, 253, 253, 250, 182,   0,   0,   0,   0,   0,   0,
                   0,   0],
                [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,  24, 114, 221,
                 253, 253, 253, 253, 201,  78,   0,   0,   0,   0,   0,   0,   0,
                   0,   0],
                [  0,   0,   0,   0,   0,   0,   0,   0,  23,  66, 213, 253, 253,
                 253, 253, 198,  81,   2,   0,   0,   0,   0,   0,   0,   0,   0,
                   0,   0],
                [  0,   0,   0,   0,   0,   0,  18, 171, 219, 253, 253, 253, 253,
                 195,  80,   9,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
                   0,   0],
                [  0,   0,   0,   0,  55, 172, 226, 253, 253, 253, 253, 244, 133,
                  11,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
```

```
       0,   0],
       [  0,   0,   0,   0, 136, 253, 253, 253, 212, 135, 132,  16,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0],
       [  0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,   0,
          0,   0]], dtype=uint8)
```

In [41]:
```
X_train = X_train/255
X_test = X_test/255
```

In [42]:
```
X_train[0]
```

```
 0.99215686, 0.99215686, 0.46666667, 0.09803922, 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.17647059,
 0.72941176, 0.99215686, 0.99215686, 0.58823529, 0.10588235,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.0627451 , 0.36470588, 0.98823529, 0.99215686, 0.73333333,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        ],
[0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.        , 0.        , 0.        ,
 0.        , 0.        , 0.97647059, 0.99215686, 0.97647059,
 0.25098039, 0.        , 0.        , 0.        , 0.        ,
```

In [19]:
```
model = Sequential()

model.add(Flatten(input_shape=(28, 28)))
model.add(Dense(128, activation='relu'))
model.add(Dense(10, activation='softmax'))
```

In [22]:
```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 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 [23]:
```
model.compile(loss='sparse_categorical_crossentropy',optimizer='Adam',metrics=['accuracy']
```

In [45]:
```
model.fit(X_train, y_train, epochs=10, validation_split=0.2)
```

```
Epoch 1/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.2844 - accuracy: 0.918
7 - val_loss: 0.1530 - val_accuracy: 0.9581
Epoch 2/10
1500/1500 [==============================] - 5s 3ms/step - loss: 0.1250 - accuracy: 0.963
9 - val_loss: 0.1138 - val_accuracy: 0.9656
Epoch 3/10
1500/1500 [==============================] - 7s 4ms/step - loss: 0.0844 - accuracy: 0.974
7 - val_loss: 0.1059 - val_accuracy: 0.9688
Epoch 4/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0627 - accuracy: 0.981
5 - val_loss: 0.0923 - val_accuracy: 0.9724
Epoch 5/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0475 - accuracy: 0.985
8 - val_loss: 0.0880 - val_accuracy: 0.9756
Epoch 6/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0364 - accuracy: 0.989
2 - val_loss: 0.0831 - val_accuracy: 0.9758
Epoch 7/10
1500/1500 [==============================] - 5s 4ms/step - loss: 0.0278 - accuracy: 0.992
1 - val_loss: 0.0879 - val_accuracy: 0.9758
Epoch 8/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0233 - accuracy: 0.993
3 - val_loss: 0.0880 - val_accuracy: 0.9768
Epoch 9/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0189 - accuracy: 0.994
6 - val_loss: 0.0996 - val_accuracy: 0.9758
Epoch 10/10
1500/1500 [==============================] - 5s 4ms/step - loss: 0.0144 - accuracy: 0.996
1 - val_loss: 0.0891 - val_accuracy: 0.9773
```

Out[45]:
```
<keras.callbacks.History at 0x16e58b14910>
```

In [46]:
```
model.predict(X_test)
```

```
313/313 [==============================] - 1s 2ms/step
```

Out[46]:
```
array([[3.9149205e-07, 5.7159829e-11, 2.0275408e-07, ..., 9.9989229e-01,
        2.7472973e-07, 9.5624898e-07],
       [7.5929983e-12, 1.2326724e-07, 9.9999958e-01, ..., 2.3111235e-15,
        2.1620771e-07, 6.0389905e-14],
       [7.2400425e-08, 9.9963927e-01, 5.9094244e-05, ..., 2.0802880e-04,
        9.0360016e-05, 1.5908670e-07],
       ...,
       [1.9186707e-18, 1.9581691e-12, 9.9627130e-18, ..., 2.2416985e-11,
        7.5827686e-12, 1.3581136e-08],
       [7.8854614e-14, 1.0624365e-11, 4.7876933e-18, ..., 4.6557275e-11,
        1.2339046e-06, 3.9062105e-13],
       [6.0933295e-13, 1.0817684e-13, 1.4911072e-11, ..., 4.1554080e-17,
        8.8335745e-12, 4.4684949e-14]], dtype=float32)
```

In [47]:
```
y_prob = model.predict(X_test)
```

```
313/313 [==============================] - 2s 5ms/step
```

In [48]:
```
y_prob
```

Out[48]:
```
array([[3.9149205e-07, 5.7159829e-11, 2.0275408e-07, ..., 9.9989229e-01,
        2.7472973e-07, 9.5624898e-07],
       [7.5929983e-12, 1.2326724e-07, 9.9999958e-01, ..., 2.3111235e-15,
        2.1620771e-07, 6.0389905e-14],
       [7.2400425e-08, 9.9963927e-01, 5.9094244e-05, ..., 2.0802880e-04,
        9.0360016e-05, 1.5908670e-07],
       ...,
       [1.9186707e-18, 1.9581691e-12, 9.9627130e-18, ..., 2.2416985e-11,
        7.5827686e-12, 1.3581136e-08],
       [7.8854614e-14, 1.0624365e-11, 4.7876933e-18, ..., 4.6557275e-11,
        1.2339046e-06, 3.9062105e-13],
       [6.0933295e-13, 1.0817684e-13, 1.4911072e-11, ..., 4.1554080e-17,
        8.8335745e-12, 4.4684949e-14]], dtype=float32)
```

In [49]:
```
y_prob.argmax(axis=1)
```

Out[49]:
```
array([7, 2, 1, ..., 4, 5, 6], dtype=int64)
```

In [50]:
```
y_pred = y_prob.argmax(axis=1)
```

In [52]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[52]:
```
0.979
```

In [53]:
```
history = model.fit(X_train,y_train,epochs=10,validation_split=0.2)
```

```
Epoch 1/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0141 - accuracy: 0.995
6 - val_loss: 0.0963 - val_accuracy: 0.9768
Epoch 2/10
1500/1500 [==============================] - 5s 4ms/step - loss: 0.0105 - accuracy: 0.996
8 - val_loss: 0.1056 - val_accuracy: 0.9758
Epoch 3/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0101 - accuracy: 0.997
0 - val_loss: 0.1064 - val_accuracy: 0.9757
Epoch 4/10
1500/1500 [==============================] - 7s 5ms/step - loss: 0.0079 - accuracy: 0.997
9 - val_loss: 0.1057 - val_accuracy: 0.9766
Epoch 5/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0075 - accuracy: 0.997
9 - val_loss: 0.1099 - val_accuracy: 0.9757
Epoch 6/10
1500/1500 [==============================] - 7s 5ms/step - loss: 0.0079 - accuracy: 0.997
6 - val_loss: 0.1202 - val_accuracy: 0.9740
Epoch 7/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0048 - accuracy: 0.998
6 - val_loss: 0.1354 - val_accuracy: 0.9715
Epoch 8/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0076 - accuracy: 0.997
7 - val_loss: 0.1157 - val_accuracy: 0.9744
Epoch 9/10
1500/1500 [==============================] - 7s 4ms/step - loss: 0.0043 - accuracy: 0.998
8 - val_loss: 0.1155 - val_accuracy: 0.9778
Epoch 10/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0041 - accuracy: 0.998
8 - val_loss: 0.1172 - val_accuracy: 0.9768
```

In [54]:
```
y_prob = model.predict(X_test)
```

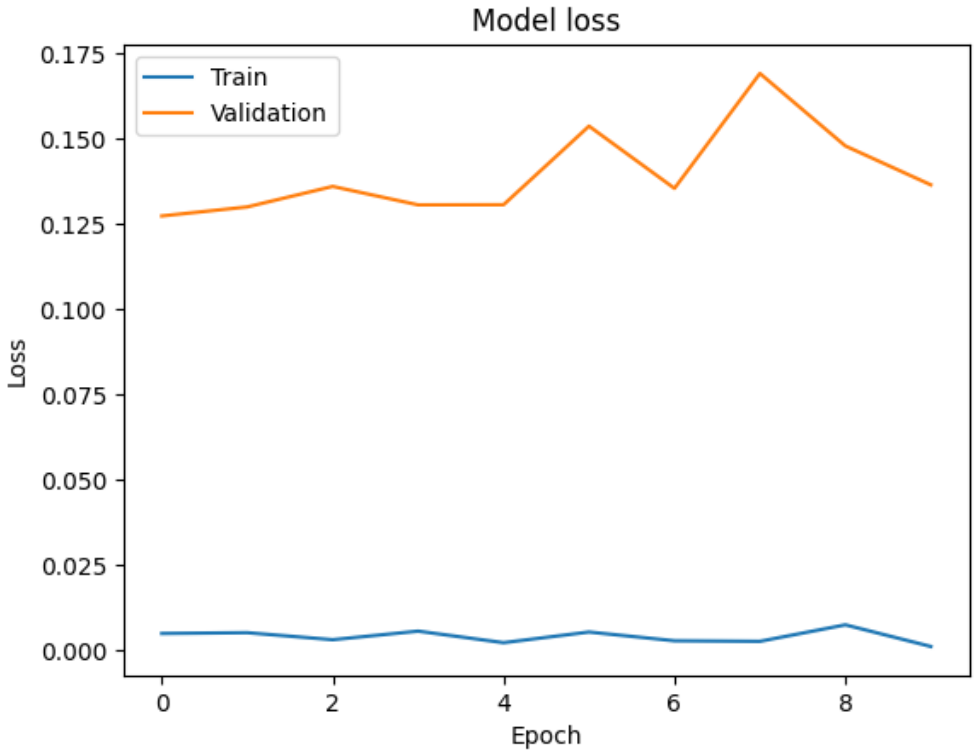
```
313/313 [==============================] - 1s 2ms/step
```

In [55]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Out[55]:
```
0.979
```

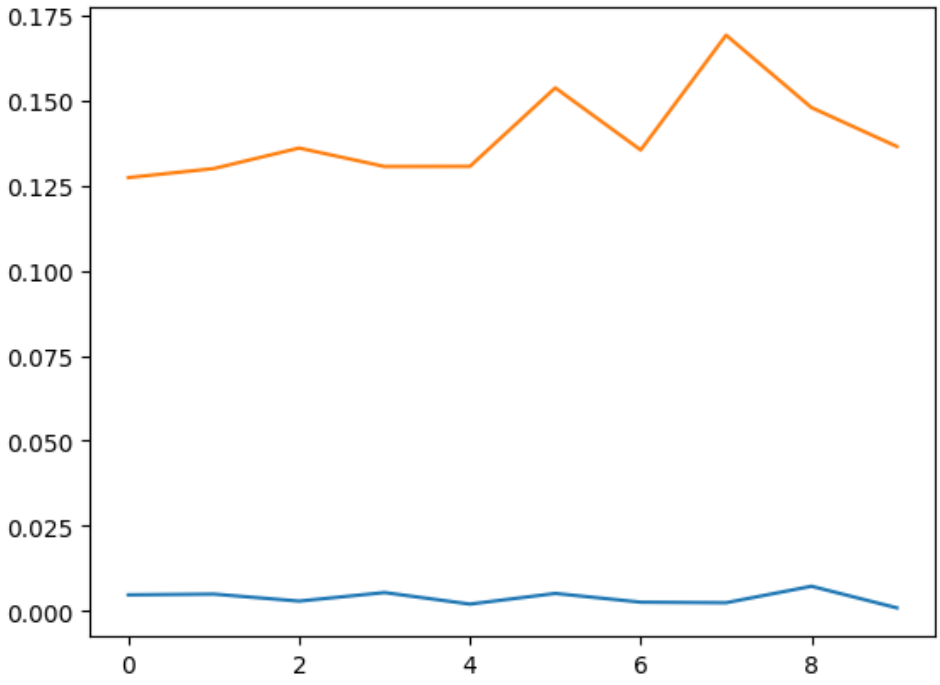
In [58]:
```
import matplotlib.pyplot as plt

# Assuming 'history' is the object returned by model.fit
history = model.fit(X_train, y_train, epochs=10, validation_split=0.2)

# Plot training & validation loss values
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Validation'], loc='upper left')
plt.show()
```

```
Epoch 1/10
1500/1500 [==============================] - 8s 5ms/step - loss: 0.0048 - accuracy: 0.998
6 - val_loss: 0.1273 - val_accuracy: 0.9763
Epoch 2/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0050 - accuracy: 0.998
5 - val_loss: 0.1299 - val_accuracy: 0.9762
Epoch 3/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0029 - accuracy: 0.999
1 - val_loss: 0.1360 - val_accuracy: 0.9759
Epoch 4/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0054 - accuracy: 0.998
2 - val_loss: 0.1305 - val_accuracy: 0.9775
Epoch 5/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0021 - accuracy: 0.999
4 - val_loss: 0.1306 - val_accuracy: 0.9765
Epoch 6/10
1500/1500 [==============================] - 7s 5ms/step - loss: 0.0052 - accuracy: 0.998
2 - val_loss: 0.1536 - val_accuracy: 0.9750
Epoch 7/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0026 - accuracy: 0.999
2 - val_loss: 0.1354 - val_accuracy: 0.9778
Epoch 8/10
1500/1500 [==============================] - 7s 5ms/step - loss: 0.0025 - accuracy: 0.999
6 - val_loss: 0.1691 - val_accuracy: 0.9708
Epoch 9/10
1500/1500 [==============================] - 6s 4ms/step - loss: 0.0073 - accuracy: 0.997
6 - val_loss: 0.1478 - val_accuracy: 0.9762
Epoch 10/10
1500/1500 [==============================] - 6s 4ms/step - loss: 9.5227e-04 - accuracy:
0.9999 - val_loss: 0.1364 - val_accuracy: 0.9797
```



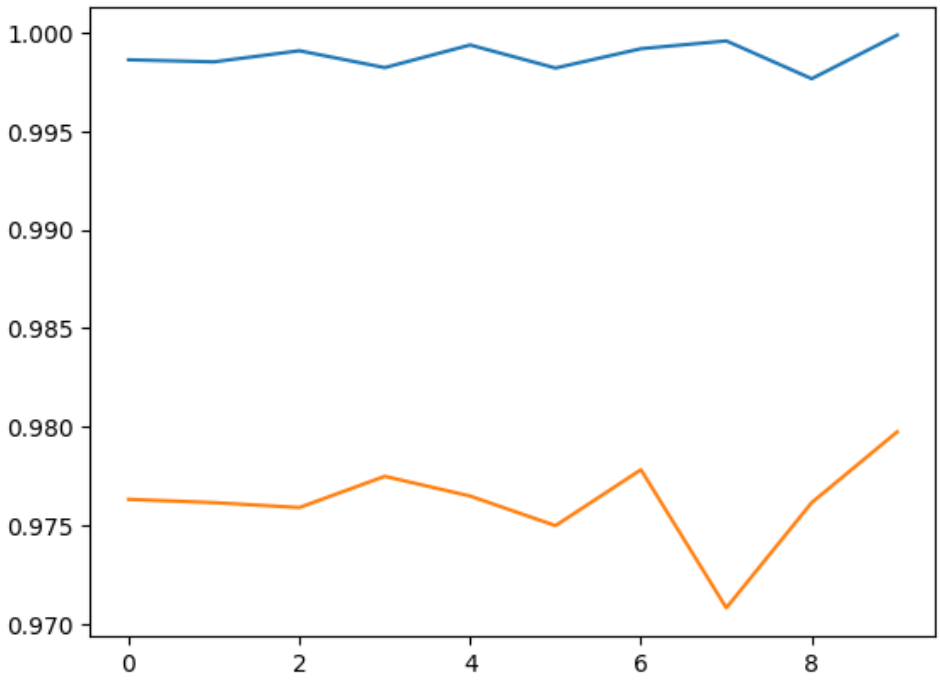
In [59]:
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```

Out[59]: [<matplotlib.lines.Line2D at 0x16e51e5efd0>]



In [60]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
```

Out[60]:
```
[<matplotlib.lines.Line2D at 0x16e515b16a0>]
```



In [61]:
```
X_test
```

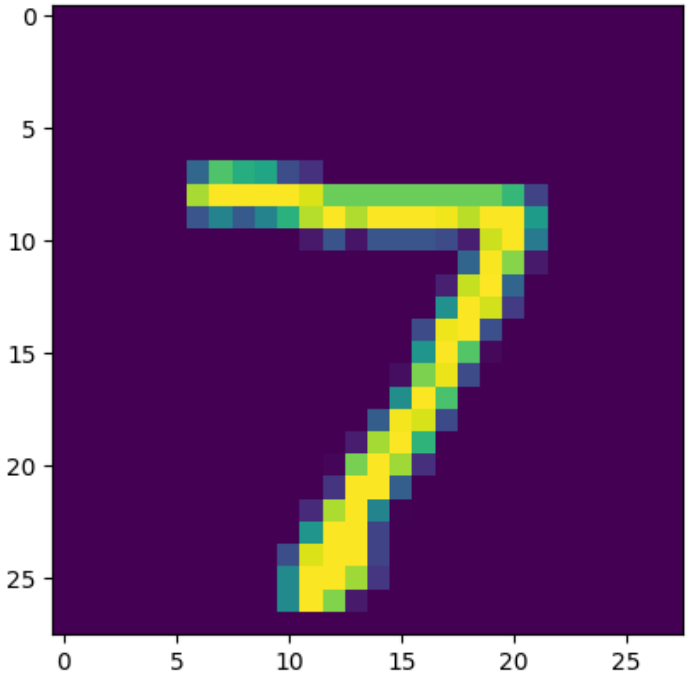
Out[61]:
```
array([[[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        ...,
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]],

       [[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 [62]:
```
plt.imshow
```

Out[62]:
```
<function matplotlib.pyplot.imshow(X: 'ArrayLike | PIL.Image.Image', cmap: 'str | Colorma
p | None' = None, norm: 'str | Normalize | None' = None, *, aspect: "Literal['equal', 'au
to'] | float | None" = None, interpolation: 'str | None' = None, alpha: 'float | ArrayLik
e | None' = None, vmin: 'float | None' = None, vmax: 'float | None' = None, origin: "Lite
ral['upper', 'lower'] | None" = None, extent: 'tuple[float, float, float, float] | None'
= None, interpolation_stage: "Literal['data', 'rgba'] | None" = None, filternorm: 'bool'
= True, filterrad: 'float' = 4.0, resample: 'bool | None' = None, url: 'str | None' = Non
e, data=None, **kwargs) -> 'AxesImage'>
```

In [63]:
```
plt.imshow(X_test[0])
```

Out[63]:
```
<matplotlib.image.AxesImage at 0x16e515e7a00>
```



In [64]:
```
model.predict(X_test[0].reshape(1,28,28))
```

```
1/1 [==============================] - 0s 90ms/step
```

Out[64]:
```
array([[1.6196361e-13, 4.2471322e-21, 6.3060607e-14, 1.3281119e-07,
        2.1085243e-27, 5.6176652e-18, 7.5485015e-26, 9.9999988e-01,
        9.2970537e-16, 1.2134247e-12]], dtype=float32)
```

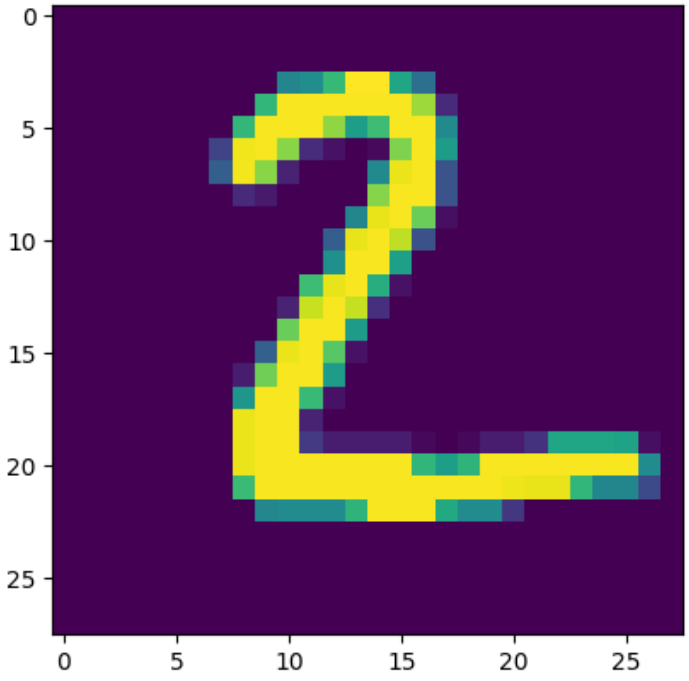
In [65]:
```
model.predict(X_test[0].reshape(1,28,28)).argmax(axis=1)
```

```
1/1 [==============================] - 0s 93ms/step
```

Out[65]:
```
array([7], dtype=int64)
```

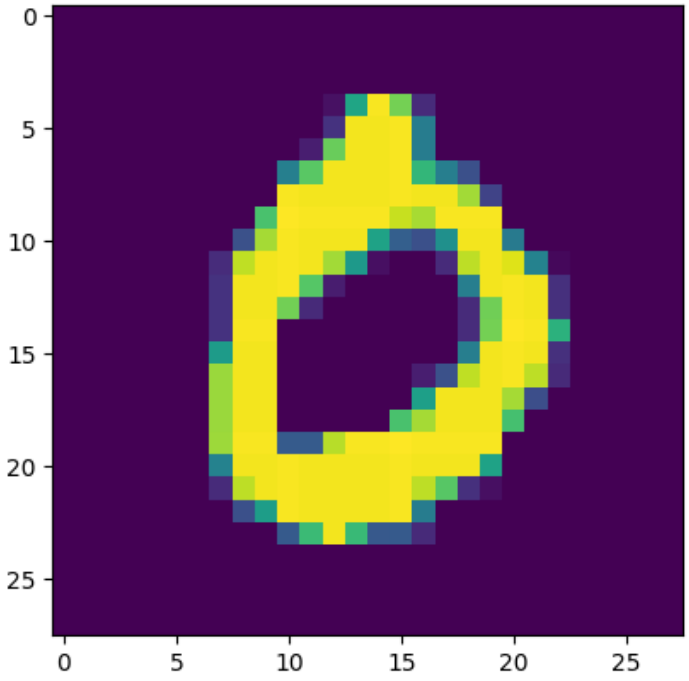
In [66]:
```
plt.imshow(X_test[1])
```

Out[66]: <matplotlib.image.AxesImage at 0x16e51eb72b0>



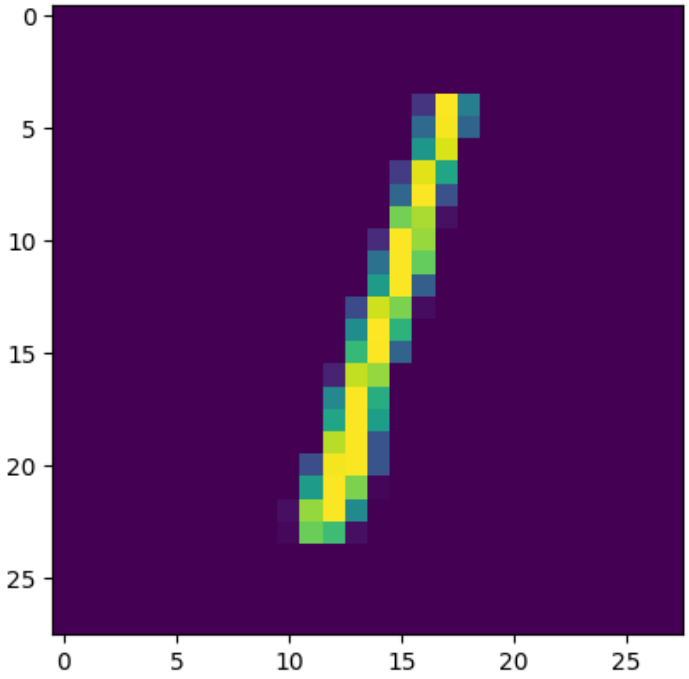
In [67]:
```
plt.imshow(X_test[3])
```

Out[67]: <matplotlib.image.AxesImage at 0x16e51e9a3d0>



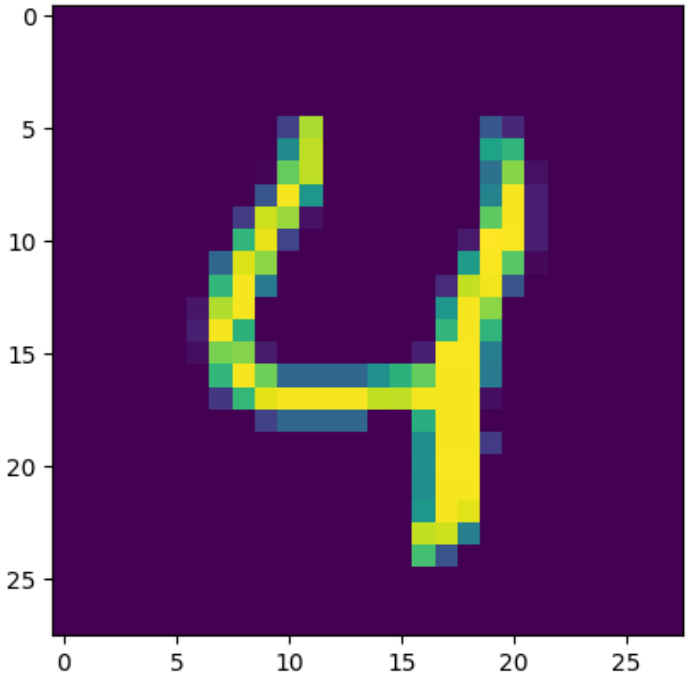
In [68]:
```
plt.imshow(X_test[2])
```

Out[68]:
```
<matplotlib.image.AxesImage at 0x16e5308d190>
```



In [69]:
```
plt.imshow(X_test[4])
```

Out[69]:
```
<matplotlib.image.AxesImage at 0x16e51f91e80>
```



In [70]:
```
model.predict(X_test[1].reshape(1,28,28))
```

```
1/1 [==============================] - 0s 93ms/step
```

Out[70]:
```
array([[4.2280255e-21, 9.4029645e-13, 1.0000000e+00, 4.5929679e-19,
        5.1775389e-36, 4.2237878e-20, 1.9520854e-25, 4.2119833e-25,
        1.2536831e-15, 9.5664205e-22]], dtype=float32)
```

In [ ]: