Loading tensorflow and other required libraries

In [1]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv2D, MaxPool2D, Flatten, Dropout
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Collecting data

In [2]:
```
(x_train,y_train),(x_test, y_test) = tf.keras.datasets.mnist.load_data()
```

In [3]:
```
x_train
```

Out[3]:
```
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 [4]:
```
x_train.shape
```

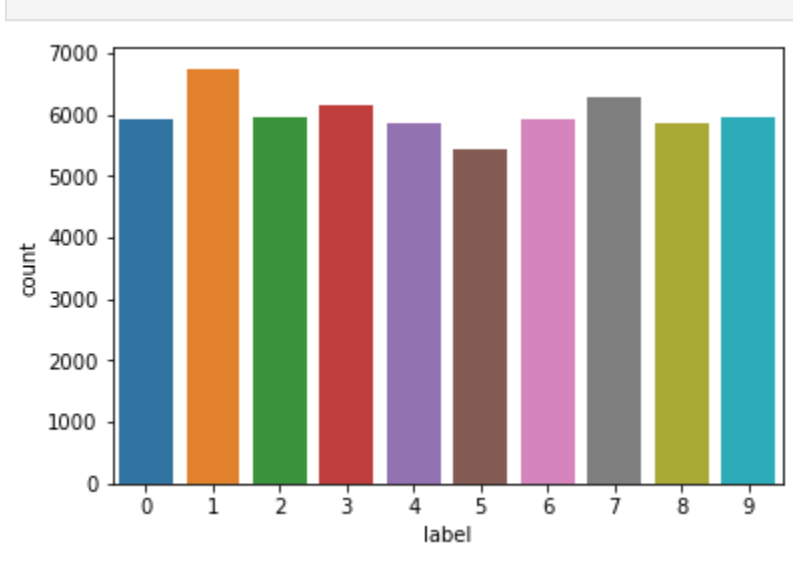
Out[4]: (60000, 28, 28)

In [5]:
```
x_test.shape
```

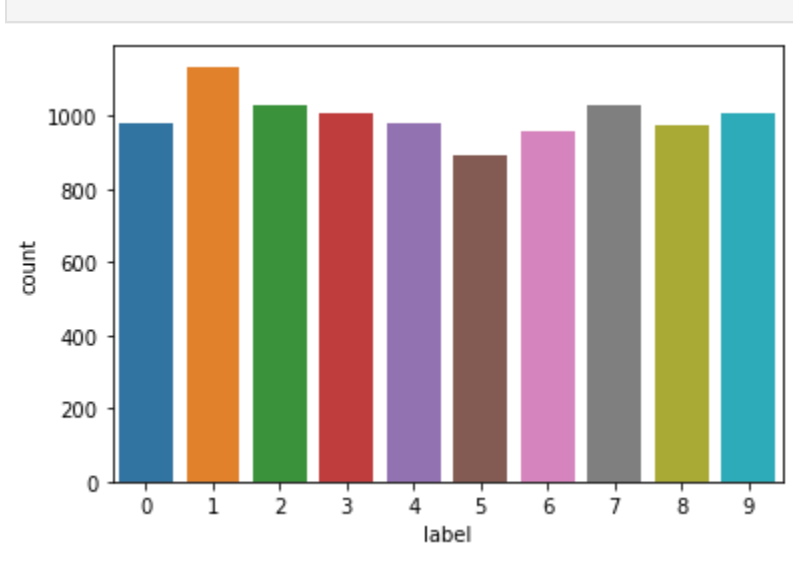
Out[5]: (10000, 28, 28)

Visualisation of occurrence of each digit in dataset

In [6]:
```
df=pd.DataFrame(y_train,columns=['label'])
sns.countplot(x=df['label'], data=df)
plt.show()
```

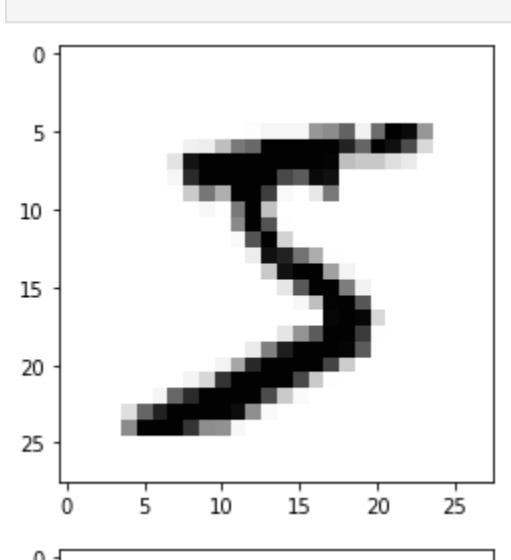
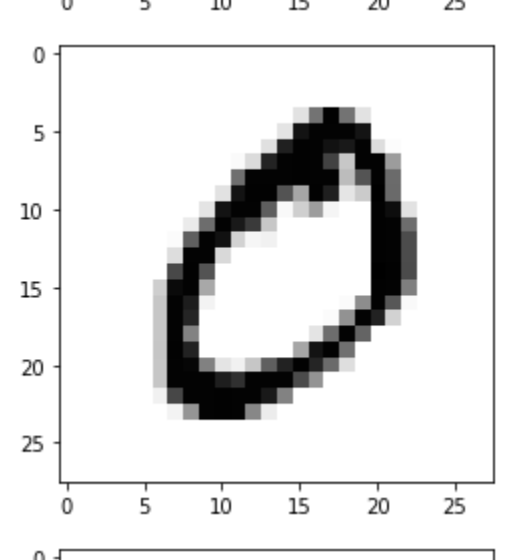
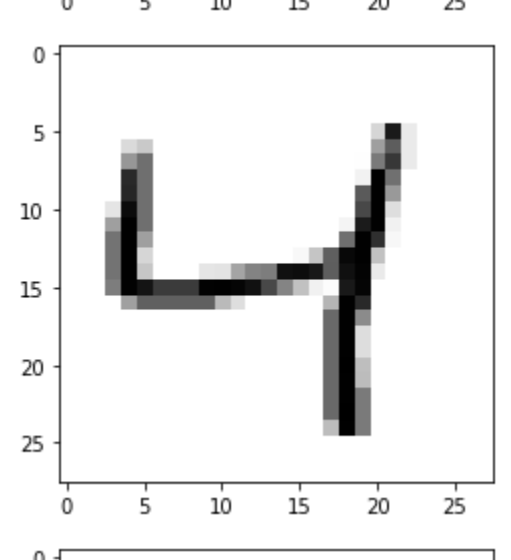
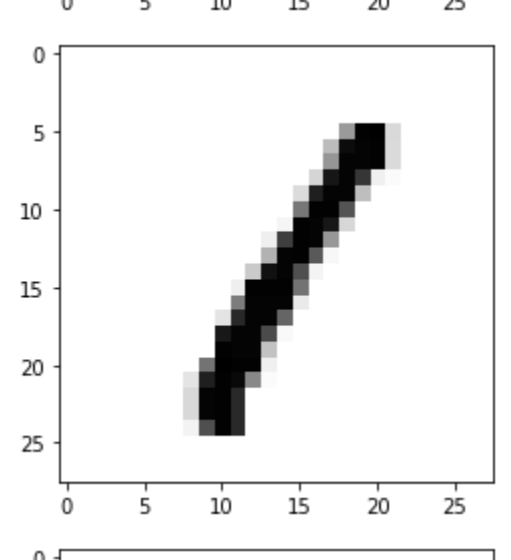
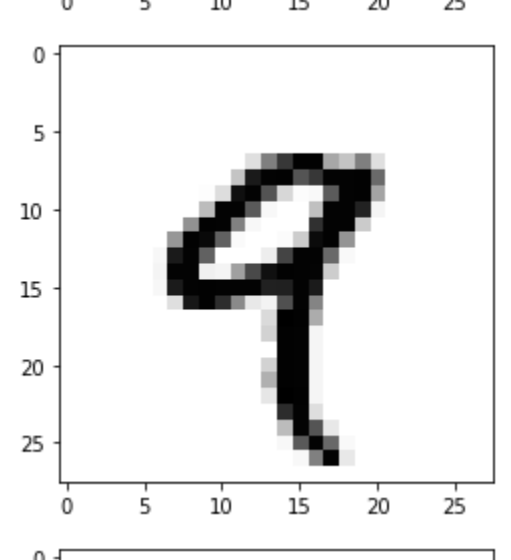
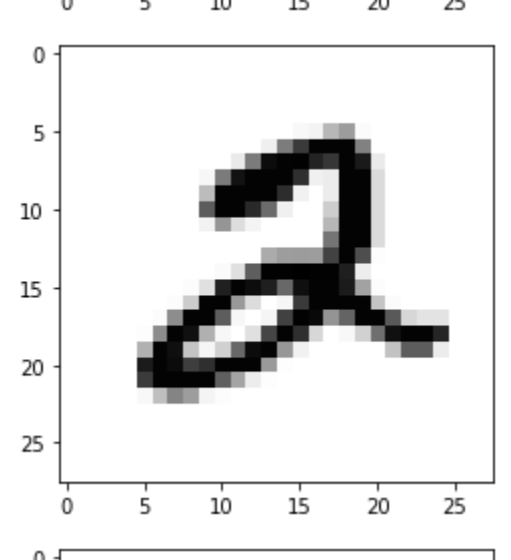
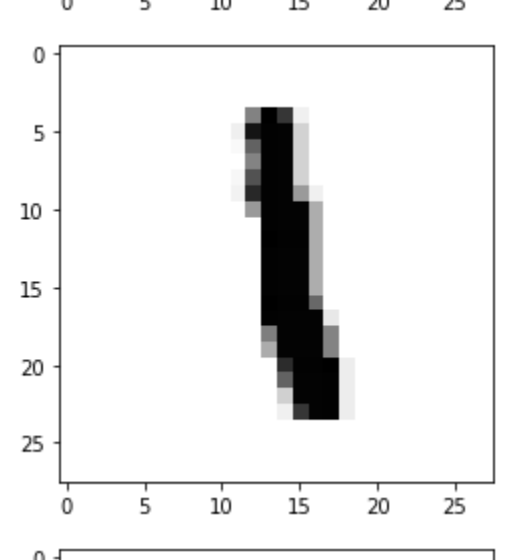
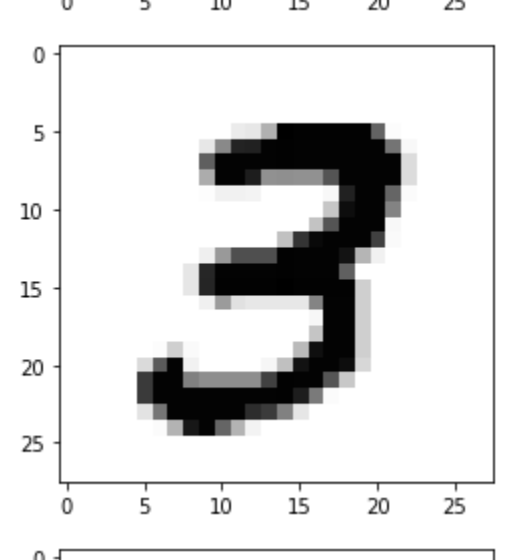
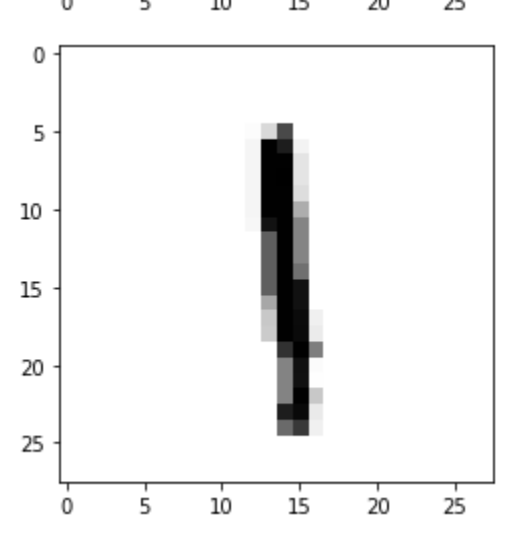
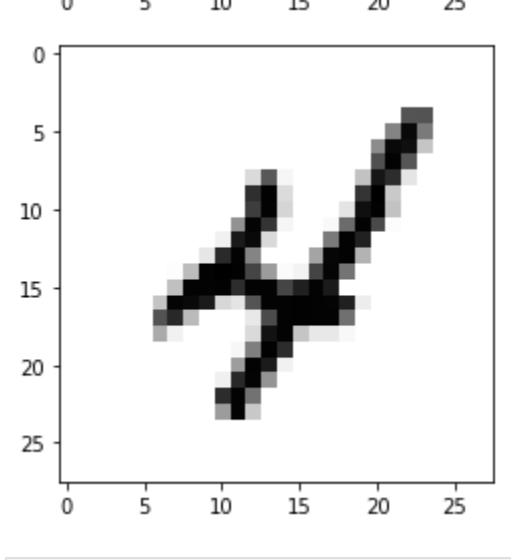


In [7]:
```
df1=pd.DataFrame(y_test,columns=['label'])
sns.countplot(x=df1['label'], data=df1)
plt.show()
```



Visualisation of first 10 images of dataset

In [8]:
```
for i in range(0,10):
    plt.imshow(x_train[i], cmap='Greys')
    plt.show()
```

In [9]:
```
y_train[0:10]
```

Out[9]: array([5, 0, 4, 1, 9, 2, 1, 3, 1, 4], dtype=uint8)

Convert output label to multiple values

In [10]:
```
y_train1 = tf.keras.utils.to_categorical(y_train, num_classes=10)
y_test1 = tf.keras.utils.to_categorical(y_test, num_classes=10)
```

In [11]:
```
#Reshaping input data to train cnn
x_train_final = x_train.reshape(60000,28,28,1)
x_test_final = x_test.reshape(10000,28,28,1)
```

Building the Graph

In [12]:
```
# Instantiate model
model = Sequential()

# Convolution layer 1
model.add(Conv2D(filters=32, kernel_size=(5,5), strides=(1,1), input_shape=(28,28,1), padding='Same', activation='relu'))
# Pooling layer (selected half of kernel_size)
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))

# Convolution layer 2
model.add(Conv2D(filters=64, kernel_size=(5,5), strides=(1,1), input_shape=(28,28,1), padding='Same', activation='relu'))
# Pooling layer
model.add(MaxPool2D(pool_size=(2,2)))
model.add(Dropout(0.25))

# Flattening image
model.add(Flatten())

# Dense layer
model.add(Dense(120, activation='relu'))
# Dense layer
model.add(Dense(60, activation='relu'))
# Output layer
model.add(Dense(10, activation='softmax'))
```

In [13]:
```
# Compiling model
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Review model

In [14]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 28, 28, 32)        832
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 14, 14, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 14, 14, 64)        51264
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 7, 7, 64)          0
_________________________________________________________________
dropout_1 (Dropout)          (None, 7, 7, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 3136)              0
_________________________________________________________________
dense (Dense)                (None, 120)               376440
_________________________________________________________________
dense_1 (Dense)              (None, 60)                7260
_________________________________________________________________
dense_2 (Dense)              (None, 10)                610
=================================================================
Total params: 436,406
Trainable params: 436,406
Non-trainable params: 0
_________________________________________________________________
```

Training the model

In [15]:
```
model.fit(x_train_final,y_train1,
          validation_data=(x_test_final,y_test1),
          epochs=5,
          batch_size=32)
```

```
Train on 60000 samples, validate on 10000 samples
Epoch 1/5
60000/60000 [==============================] - 101s 2ms/sample - loss: 0.2957 - accuracy: 0.9336 - val_loss: 0.0588 - val_accuracy: 0.9820
Epoch 2/5
60000/60000 [==============================] - 92s 2ms/sample - loss: 0.0900 - accuracy: 0.9735 - val_loss: 0.0966 - val_accuracy: 0.9702
Epoch 3/5
60000/60000 [==============================] - 87s 1ms/sample - loss: 0.0764 - accuracy: 0.9778 - val_loss: 0.0478 - val_accuracy: 0.9849
Epoch 4/5
60000/60000 [==============================] - 86s 1ms/sample - loss: 0.0685 - accuracy: 0.9796 - val_loss: 0.0375 - val_accuracy: 0.9875
Epoch 5/5
60000/60000 [==============================] - 92s 2ms/sample - loss: 0.0641 - accuracy: 0.9818 - val_loss: 0.0340 - val_accuracy: 0.9908
```

Out[15]: <tensorflow.python.keras.callbacks.History at 0x1c01e3cb908>

Testing the model

In [16]:
```
model.evaluate(x_test_final,y_test1)
```

```
10000/10000 [==============================] - 3s 304us/sample - loss: 0.0340 - accuracy: 0.9908
```

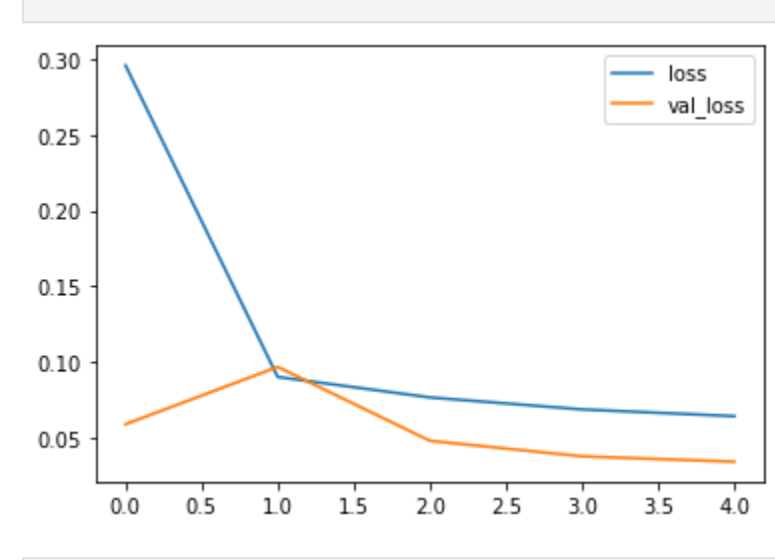
Out[16]: [0.033998030390345955, 0.9908]

In [17]:
```
metrics = pd.DataFrame(model.history.history)
metrics
```

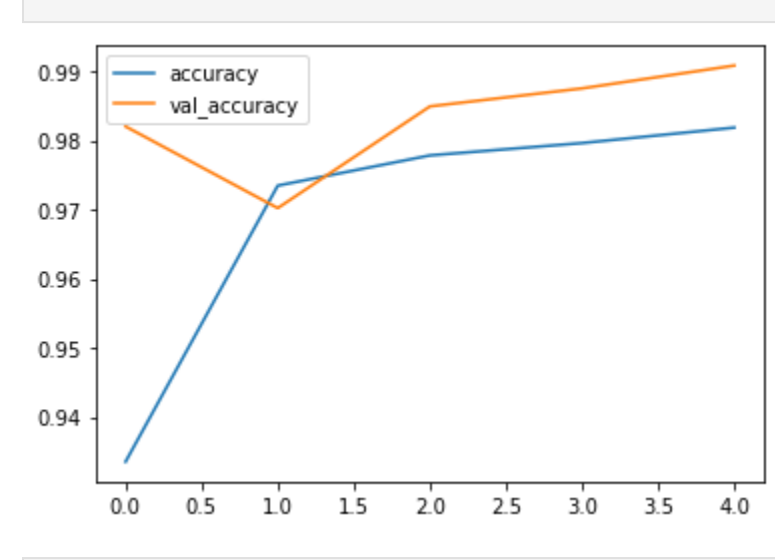
Out[17]:

	loss	accuracy	val_loss	val_accuracy
0	0.295693	0.933550	0.058760	0.9820
1	0.089956	0.973450	0.096612	0.9702
2	0.076444	0.977817	0.047804	0.9849
3	0.068518	0.979600	0.037528	0.9875
4	0.064075	0.981833	0.033998	0.9908

In [18]:
```
metrics[['loss', 'val_loss']].plot()
plt.show()
```



In [22]:
```
metrics[['accuracy', 'val_accuracy']].plot()
plt.show()
```



In [23]:
```
y_pred=model.predict_classes(x_test_final)
```

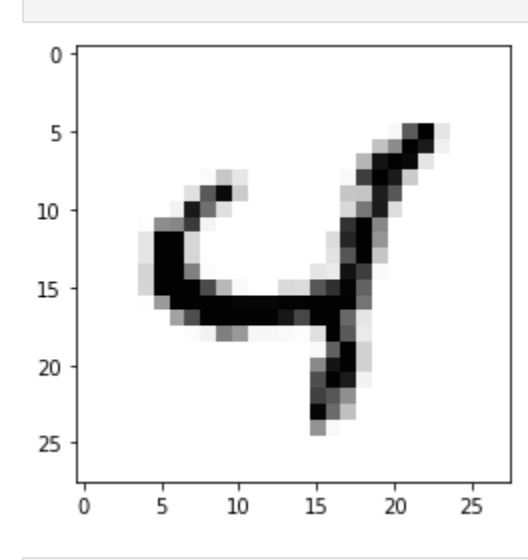
In [24]:
```
y_pred
```

Out[24]: array([7, 2, 1, ..., 4, 5, 6], dtype=int64)

In [25]:
```
from sklearn.metrics import classification_report, confusion_matrix
print('Classification Report:')
print(classification_report(y_test, y_pred))
print('\n')
print('Confusion Matrix:')
print(confusion_matrix(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       980
           1       0.99      1.00      1.00      1135
           2       0.99      1.00      0.99      1032
           3       0.99      0.99      0.99      1010
           4       0.99      0.99      0.99       982
           5       1.00      0.98      0.99       892
           6       0.99      0.99      0.99       958
           7       0.99      0.99      0.99      1028
           8       0.99      0.99      0.99       974
           9       0.99      0.98      0.98      1009

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000



Confusion Matrix:
[[ 976    0    0    0    0    0    3    1    0    0]
 [   0 1131    1    1    1    0    0    0    1    0]
 [   0    0 1027    0    1    0    0    4    0    0]
 [   0    0    1 1001    0    1    0    2    3    2]
 [   0    0    0    0  976    0    0    1    1    4]
 [   1    0    0   10    1  874    2    0    1    3]
 [   1    3    2    0    3    1  946    0    2    0]
 [   0    2    2    0    0    0    0 1021    1    2]
 [   1    0    2    1    1    0    0    2  966    1]
 [   1    2    0    2    7    2    0    4    1  990]]
```

In [32]:
```
plt.imshow(x_test[67], cmap='Greys')
```

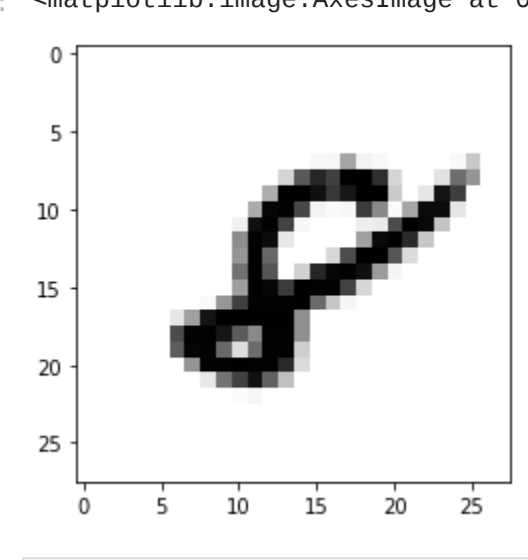


In [31]:
```
y_pred[67]
```

Out[31]: 4

In [34]:
```
plt.imshow(x_test[6725], cmap='Greys')
```

Out[34]: <matplotlib.image.AxesImage at 0x1c02190e348>

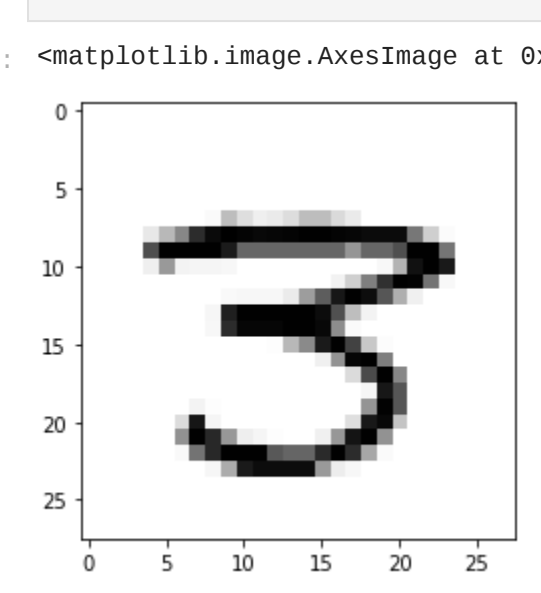


In [35]:
```
y_pred[6725]
```

Out[35]: 8

In [36]:
```
plt.imshow(x_test[3456], cmap='Greys')
```

Out[36]: <matplotlib.image.AxesImage at 0x1c021978548>

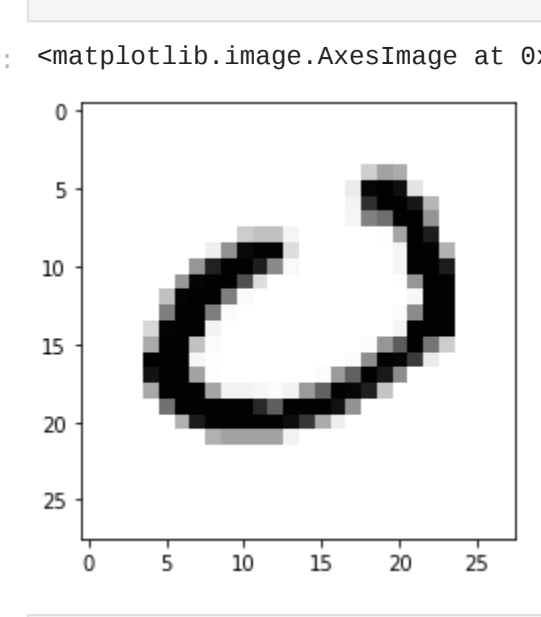


In [38]:
```
y_pred[3456]
```

Out[38]: 3

In [45]:
```
plt.imshow(x_test[8212], cmap='Greys')
```

Out[45]: <matplotlib.image.AxesImage at 0x1c021d024c8>

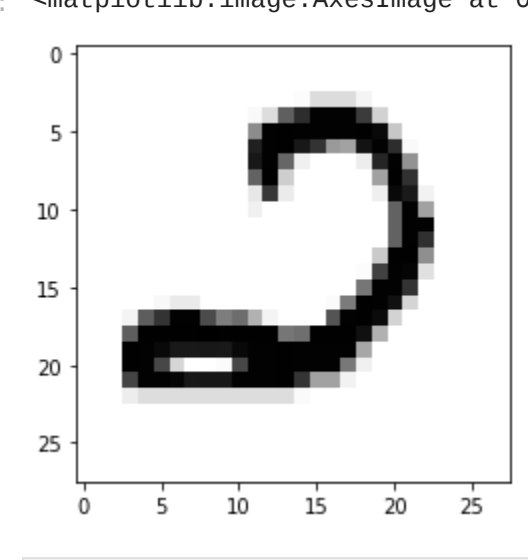


In [46]:
```
y_pred[8212]
```

Out[46]: 0

In [52]:
```
plt.imshow(x_test[9421], cmap='Greys')
```

Out[52]: <matplotlib.image.AxesImage at 0x1c021895408>



In [53]:
```
y_pred[9421]
```

Out[53]: 2

In [55]:
```
model.save("my_cnn_model")
```

```
WARNING:tensorflow:From C:\Users\manis\anaconda3\envs\tensor\lib\site-packages\tensorflow_core\python\ops\resource_variable_ops.py:1786: calling BaseResourceVariable.__init__ (from tensorflow.python.ops.resource_variable_ops) with constraint is deprecated and will be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
INFO:tensorflow:Assets written to: my_cnn_model\assets
```

In [ ]: